Better Trade Screen

Extensions extracted

.md
.xml
.modinfo
.txt
.lua
.sample

Project: Better Trade Screen

Table of Contents ToC Page: 1

```
<?xml version="1.0" encoding="utf-8"?>
<Mod id="8d4fa23a-ef43-440c-8422-2bec11f8f5d7"  version="4.2">
  <Properties>
    <Name>Better Trade Screen</Name>
    <Stability>Beta</Stability>
    <Teaser>Adds sort options, filters and generally improves the trade
screen.</Teaser>
    <Description>Adds sort options, filters and generally improves the trade
screen.</Description>
    <Authors>astog</Authors>
    <AffectsSavedGames>0</AffectsSavedGames>
    <CompatibleVersions>2.0</CompatibleVersions>
  </Properties>
  <InGameActions>
    <ImportFiles id="BTS_IMPORT_FILES">
      <Items>
        <File>UI/TradeOverview.xml</File>
        <File>UI/TradeOverview.lua</File>
        <File>UI/TradeSupport.lua</File>
        <File>UI/Choosers/TradeRouteChooser.xml</File>
        <File>UI/Choosers/TradeRouteChooser.lua</File>
        <File>UI/Choosers/TradeOriginChooser.xml</File>
        <File>UI/Choosers/TradeOriginChooser.lua</File>
      </Items>
    </ImportFiles>
    <LocalizedText id="BTS_TEXT">
      <Items>
        <File>Text/BTS_Text_DE.xml</File>
        <File>Text/BTS_Text_EN.xml</File>
        <File>Text/BTS_Text_ES.xml</File>
        <File>Text/BTS_Text_FR.xml</File>
        <File>Text/BTS_Text_Hans_CN.xml</File>
        <File>Text/BTS_Text_IT.xml</File>
        <File>Text/BTS_Text_KR.xml</File>
        <File>Text/BTS_Text_PL.xml</File>
        <File>Text/BTS_Text_RU.xml</File>
      </Items>
    </LocalizedText>
  </InGameActions>
  <Files>
    <File>UI/TradeOverview.xml</File>
    <File>UI/TradeOverview.lua</File>
    <File>UI/TradeSupport.lua</File>
```

```
    <File>UI/Choosers/TradeRouteChooser.xml</File>
    <File>UI/Choosers/TradeRouteChooser.lua</File>
    <File>UI/Choosers/TradeOriginChooser.xml</File>
    <File>UI/Choosers/TradeOriginChooser.lua</File>
    <File>Text/BTS_Text_DE.xml</File>
    <File>Text/BTS_Text_EN.xml</File>
    <File>Text/BTS_Text_ES.xml</File>
    <File>Text/BTS_Text_FR.xml</File>
    <File>Text/BTS_Text_Hans_CN.xml</File>
    <File>Text/BTS_Text_IT.xml</File>
    <File>Text/BTS_Text_KR.xml</File>
    <File>Text/BTS_Text_PL.xml</File>
    <File>Text/BTS_Text_RU.xml</File>
  </Files>
</Mod>
```

```
# Better Trade Screen
## Introduction
The goal of this mod is to improve the trade screens in Civilization VI and help manage and monitor running trade routes.
## Features
* Shows turns to complete a trade route rather than the distance between the cities.
* Overhauled *Trade Overview* screen.
        * Shows all possible routes, even if the trader is not present in the origin city.
        * Clicking on a route where a free trade unit is not present in the origin city takes you to a free trade unit and opens the *Change City* screen
        * Route entry is colored based on destination player.
        * Player/City header are also colored.
        * Shows origin city and destination city yields in the same screen.
        * Added **Group** and **Filter** settings
        * *My Routes* tab tracks active routes, so you know when a trade route completes.
        ![alt text](http://i.imgur.com/3G1PAdh.jpg?1 "Overhauled Trade Overview screen")
* Sort bar in *Make Trade Route* screen and *Trade Overview* screen. Sort the routes by **left clicking** on a button.
        ![alt text](http://i.imgur.com/QUTDQYe.jpg "Sort bar - Trade Overview")
* Trade Routes can be sorted based on yields, and turns remaining. Queue multiple sorts by holding **SHIFT** and the left clicking on a sort button. Right click on any sort button to remove it from the sort setting.
        ![alt text](http://i.imgur.com/C1T7kPL.jpg?1 "Multiple Sort example")
* When opening *Make Trade Route* screen, the last destination is automatically picked.
* Set a trader to repeat its last route by selecting the **Repeat Route** checkbox when initiating a trade route.
        ![alt text](http://i.imgur.com/faLa0b3.jpg "Repeat Route checkbox")
* An additional checkbox is provided that sets the trader to repeat the **top** route from the sort settings when the trade was initiated. This allows the trade route to always be the best one, hence reducing micromanagent of always checking the trade routes.
* Cancel the automated trader from the *My Routes* tab in **Trade Overview** screen.
## Installation
If you are using [Chao's QUI](https://github.com/chaorace/cqui), this mod is already included in it, and requires no extra steps to install. If you are **NOT** using CQUI, follow the steps below:
1. Download the latest release.
```

2. Extract the downloaded archive to your Mods folder. For me this is in *Documents\My Games\Sid Meier's Civilization VI\Mods*
3. Activate the Mod in *Additional Content* inside Civilization VI.
Troubleshooting
If you encounter issues with getting the mod working try the following steps:
1. Try installing the Mod into the DLC folder. This folder is the folder where you installed Civilization VI, example *C:\Program Files (x86)\Steam\steamapps\common\Sid Meier's Civilization VI\DLC*
2. Delete the cache. This can be found here - *Documents\My Games\Sid Meier's Civilization VI\Cache*
3. Check out this [thread](https://forums.civfanatics.com/threads/mods-not-working-at-all-help.606288/)
4. If none of the above work, let me know in this repository or [here](https://forums.civfanatics.com/threads/more-lenses.606150/)
FAQ
I loaded a save game and the text is all broken?
> This is a bug from Firaxis. To fix this you have to exit to **destop** and start Civilization VI again.
I can't see yields for the destination city, where are they?
>In the previous version, I had a string show that the destination city gains no benefits, but it lead to a lot of cluttering in the screen. Currently, if the destination city has no yield, you won't see any.
Trade Overview panel does not open in between turns?
>With v3.0 I blocked the Trade Overview panel from opening since it causes CTD. If you want to unblock this, change the following line in TradeOverview.lua
>
>`local blockPanelInBetweenTurns = true`
>
>to
>
>`local blockPanelInBetweenTurns = false`
Credits
* @ZhouYzzz for providing the Chinese localization in #161-CQUI
* @deggesim (Simone1974 on Civfanatics) for providing the Italian localization in #250-CQUI
* @e1ectron for providing the Russian localization in #251-CQUI
* @sejbr for providing the Polish localization in #253-CQUI
* @frytom for providing the German localization in #283-CQUI
* @lctrs for providing a partial French localization in #273-CQUI
* @wbqd for providing a Korean translation in #309-CQUI

```
#!/bin/sh
#
# An example hook script to check the commit log message taken by
# applypatch from an e-mail message.
#
# The hook should exit with non-zero status after issuing an
# appropriate message if it wants to stop the commit.  The hook is
# allowed to edit the commit message file.
#
# To enable this hook, rename this file to "applypatch-msg".
. git-sh-setup
commitmsg="$(git rev-parse --git-path hooks/commit-msg)"
test -x "$commitmsg" && exec "$commitmsg" ${1+"$@"}
:
```

```
#!/bin/sh
#
# An example hook script to check the commit log message.
# Called by "git commit" with one argument, the name of the file
# that has the commit message.  The hook should exit with non-zero
# status after issuing an appropriate message if it wants to stop the
# commit.  The hook is allowed to edit the commit message file.
#
# To enable this hook, rename this file to "commit-msg".
# Uncomment the below to add a Signed-off-by line to the message.
# Doing this in a hook is a bad idea in general, but the prepare-commit-msg
# hook is more suited to it.
#
# SOB=$(git var GIT_AUTHOR_IDENT | sed -n 's/^\(.*>\).*$/Signed-off-by: \1/p')
# grep -qs "^$SOB" "$1" || echo "$SOB" >> "$1"
# This example catches duplicate Signed-off-by lines.
test "" = "$(grep '^Signed-off-by: ' "$1" |
         sort | uniq -c | sed -e '/^[   ]*1[    ]/d')" || {
        echo >&2 Duplicate Signed-off-by lines.
        exit 1
}
```

```
#!/usr/bin/perl
use strict;
use warnings;
use IPC::Open2;
# An example hook script to integrate Watchman
# (https://facebook.github.io/watchman/) with git to speed up detecting
# new and modified files.
#
# The hook is passed a version (currently 1) and a time in nanoseconds
# formatted as a string and outputs to stdout all files that have been
# modified since the given time. Paths must be relative to the root of
# the working tree and separated by a single NUL.
#
# To enable this hook, rename this file to "query-watchman" and set
# 'git config core.fsmonitor .git/hooks/query-watchman'
#
my ($version, $time) = @ARGV;
# Check the hook interface version
if ($version == 1) {
        # convert nanoseconds to seconds
        $time = int $time / 1000000000;
} else {
        die "Unsupported query-fsmonitor hook version '$version'.\n" .
            "Falling back to scanning...\n";
}
my $git_work_tree;
if ($^O =~ 'msys' || $^O =~ 'cygwin') {
        $git_work_tree = Win32::GetCwd();
        $git_work_tree =~ tr/\\/\//;
} else {
        require Cwd;
        $git_work_tree = Cwd::cwd();
}
my $retry = 1;
launch_watchman();
sub launch_watchman {
        my $pid = open2(\*CHLD_OUT, \*CHLD_IN, 'watchman -j --no-pretty')
            or die "open2() failed: $!\n" .
            "Falling back to scanning...\n";
        # In the query expression below we're asking for names of files that
        # changed since $time but were not transient (ie created after
        # $time but no longer exist).
        #
```

```
        # To accomplish this, we're using the "since" generator to use the
        # recency index to select candidate nodes and "fields" to limit the
        # output to file names only. Then we're using the "expression" term to
        # further constrain the results.
        #
        # The category of transient files that we want to ignore will have a
        # creation clock (cclock) newer than $time_t value and will also not
        # currently exist.
        my $query = <<" END";
                ["query", "$git_work_tree", {
                        "since": $time,
                        "fields": ["name"],
                        "expression": ["not", ["allof", ["since", $time,
"cclock"], ["not", "exists"]]]
                }]
        END
        print CHLD_IN $query;
        close CHLD_IN;
        my $response = do {local $/; <CHLD_OUT>};
        die "Watchman: command returned no output.\n" .
            "Falling back to scanning...\n" if $response eq "";
        die "Watchman: command returned invalid output: $response\n" .
            "Falling back to scanning...\n" unless $response =~ /^\{/;
        my $json_pkg;
        eval {
                require JSON::XS;
                $json_pkg = "JSON::XS";
                1;
        } or do {
                require JSON::PP;
                $json_pkg = "JSON::PP";
        };
        my $o = $json_pkg->new->utf8->decode($response);
        if ($retry > 0 and $o->{error} and $o->{error} =~ m/unable to resolve
root .* directory (.*) is not watched/) {
                print STDERR "Adding '$git_work_tree' to watchman's watch
list.\n";
                $retry--;
                qx/watchman watch "$git_work_tree"/;
                die "Failed to make watchman watch '$git_work_tree'.\n" .
                    "Falling back to scanning...\n" if $? != 0;
                # Watchman will always return all files on the first query so
                # return the fast "everything is dirty" flag to git and do the
```

```
                # Watchman query just to get it over with now so we won't pay
                # the cost in git to look up each individual file.
                print "/\0";
                eval { launch_watchman() };
                exit 0;
        }
        die "Watchman: $o->{error}.\n" .
            "Falling back to scanning...\n" if $o->{error};
        binmode STDOUT, ":utf8";
        local $, = "\0";
        print @{$o->{files}};
}
```

```
#!/bin/sh
#
# An example hook script to prepare a packed repository for use over
# dumb transports.
#
# To enable this hook, rename this file to "post-update".
exec git update-server-info
```

```
#!/bin/sh
#
# An example hook script to verify what is about to be committed
# by applypatch from an e-mail message.
#
# The hook should exit with non-zero status after issuing an
# appropriate message if it wants to stop the commit.
#
# To enable this hook, rename this file to "pre-applypatch".
. git-sh-setup
precommit="$(git rev-parse --git-path hooks/pre-commit)"
test -x "$precommit" && exec "$precommit" ${1+"$@"}
:
```

```
#!/bin/sh
#
# An example hook script to verify what is about to be committed.
# Called by "git commit" with no arguments.  The hook should
# exit with non-zero status after issuing an appropriate message if
# it wants to stop the commit.
#
# To enable this hook, rename this file to "pre-commit".
if git rev-parse --verify HEAD >/dev/null 2>&1
then
        against=HEAD
else
        # Initial commit: diff against an empty tree object
        against=$(git hash-object -t tree /dev/null)
fi
# If you want to allow non-ASCII filenames set this variable to true.
allownonascii=$(git config --bool hooks.allownonascii)
# Redirect output to stderr.
exec 1>&2
# Cross platform projects tend to avoid non-ASCII filenames; prevent
# them from being added to the repository. We exploit the fact that the
# printable range starts at the space character and ends with tilde.
if [ "$allownonascii" != "true" ] &&
        # Note that the use of brackets around a tr range is ok here, (it's
        # even required, for portability to Solaris 10's /usr/bin/tr), since
        # the square bracket bytes happen to fall in the designated range.
        test $(git diff --cached --name-only --diff-filter=A -z $against |
          LC_ALL=C tr -d '[ -~]\0' | wc -c) != 0
then
        cat <<\EOF
Error: Attempt to add a non-ASCII file name.
This can cause problems if you want to work with people on other platforms.
To be portable it is advisable to rename the file.
If you know what you are doing you can disable this check using:
  git config hooks.allownonascii true
EOF
        exit 1
fi
# If there are whitespace errors, print the offending file names and fail.
exec git diff-index --check --cached $against --
```

```
#!/bin/sh
# An example hook script to verify what is about to be pushed.  Called by "git
# push" after it has checked the remote status, but before anything has been
# pushed.  If this script exits with a non-zero status nothing will be pushed.
#
# This hook is called with the following parameters:
#
# $1 -- Name of the remote to which the push is being done
# $2 -- URL to which the push is being done
#
# If pushing without using a named remote those arguments will be equal.
#
# Information about the commits which are being pushed is supplied as lines to
# the standard input in the form:
#
#   <local ref> <local sha1> <remote ref> <remote sha1>
#
# This sample shows how to prevent push of commits where the log message starts
# with "WIP" (work in progress).
remote="$1"
url="$2"
z40=0000000000000000000000000000000000000000
while read local_ref local_sha remote_ref remote_sha
do
        if [ "$local_sha" = $z40 ]
        then
                # Handle delete
                :
        else
                if [ "$remote_sha" = $z40 ]
                then
                        # New branch, examine all commits
                        range="$local_sha"
                else
                        # Update to existing branch, examine new commits
                        range="$remote_sha..$local_sha"
                fi
                # Check for WIP commit
                commit=`git rev-list -n 1 --grep '^WIP' "$range"`
                if [ -n "$commit" ]
                then
                        echo >&2 "Found WIP commit in $local_ref, not pushing"
                        exit 1
```

```
                fi
        fi
done
exit 0
```

```
#!/bin/sh
#
# Copyright (c) 2006, 2008 Junio C Hamano
#
# The "pre-rebase" hook is run just before "git rebase" starts doing
# its job, and can prevent the command from running by exiting with
# non-zero status.
#
# The hook is called with the following parameters:
#
# $1 -- the upstream the series was forked from.
# $2 -- the branch being rebased (or empty when rebasing the current branch).
#
# This sample shows how to prevent topic branches that are already
# merged to 'next' branch from getting rebased, because allowing it
# would result in rebasing already published history.
publish=next
basebranch="$1"
if test "$#" = 2
then
        topic="refs/heads/$2"
else
        topic=`git symbolic-ref HEAD` ||
        exit 0 ;# we do not interrupt rebasing detached HEAD
fi
case "$topic" in
refs/heads/??/*)
        ;;
*)
        exit 0 ;# we do not interrupt others.
        ;;
esac
# Now we are dealing with a topic branch being rebased
# on top of master.  Is it OK to rebase it?
# Does the topic really exist?
git show-ref -q "$topic" || {
        echo >&2 "No such branch $topic"
        exit 1
}
# Is topic fully merged to master?
not_in_master=`git rev-list --pretty=oneline ^master "$topic"`
if test -z "$not_in_master"
then
```

```
        echo >&2 "$topic is fully merged to master; better remove it."
        exit 1 ;# we could allow it, but there is no point.
fi
# Is topic ever merged to next?  If so you should not be rebasing it.
only_next_1=`git rev-list ^master "^$topic" ${publish} | sort`
only_next_2=`git rev-list ^master           ${publish} | sort`
if test "$only_next_1" = "$only_next_2"
then
        not_in_topic=`git rev-list "^$topic" master`
        if test -z "$not_in_topic"
        then
                echo >&2 "$topic is already up to date with master"
                exit 1 ;# we could allow it, but there is no point.
        else
                exit 0
        fi
else
        not_in_next=`git rev-list --pretty=oneline ^${publish} "$topic"`
        /usr/bin/perl -e '
                my $topic = $ARGV[0];
                my $msg = "* $topic has commits already merged to public
branch:\n";
                my (%not_in_next) = map {
                        /^([0-9a-f]+) /;
                        ($1 => 1);
                } split(/\n/, $ARGV[1]);
                for my $elem (map {
                                /^([0-9a-f]+) (.*)$/;
                                [$1 => $2];
                        } split(/\n/, $ARGV[2])) {
                        if (!exists $not_in_next{$elem->[0]}) {
                                if ($msg) {
                                        print STDERR $msg;
                                        undef $msg;
                                }
                                print STDERR " $elem->[1]\n";
                        }
                }
        ' "$topic" "$not_in_next" "$not_in_master"
        exit 1
fi
<<\DOC_END
This sample hook safeguards topic branches that have been
```

```
published from being rewound.
The workflow assumed here is:
 * Once a topic branch forks from "master", "master" is never
   merged into it again (either directly or indirectly).
 * Once a topic branch is fully cooked and merged into "master",
   it is deleted.  If you need to build on top of it to correct
   earlier mistakes, a new topic branch is created by forking at
   the tip of the "master".  This is not strictly necessary, but
   it makes it easier to keep your history simple.
 * Whenever you need to test or publish your changes to topic
   branches, merge them into "next" branch.
The script, being an example, hardcodes the publish branch name
to be "next", but it is trivial to make it configurable via
$GIT_DIR/config mechanism.
With this workflow, you would want to know:
(1) ... if a topic branch has ever been merged to "next".  Young
    topic branches can have stupid mistakes you would rather
    clean up before publishing, and things that have not been
    merged into other branches can be easily rebased without
    affecting other people.  But once it is published, you would
    not want to rewind it.
(2) ... if a topic branch has been fully merged to "master".
    Then you can delete it.  More importantly, you should not
    build on top of it -- other people may already want to
    change things related to the topic as patches against your
    "master", so if you need further changes, it is better to
    fork the topic (perhaps with the same name) afresh from the
    tip of "master".
Let's look at this example:
                   o---o---o---o---o---o---o---o---o---o "next"
                  /       /           /           /
                 /   a---a---b A     /           /
                /   /               /           /
               /   /   c---c---c---c B         /
              /   /   /             \         /
             /   /   /   b---b C     \       /
            /   /   /   /             \     /
    ---o---o---o---o---o---o---o---o---o---o---o "master"
A, B and C are topic branches.
 * A has one fix since it was merged up to "next".
 * B has finished.  It has been fully merged up to "master" and "next",
   and is ready to be deleted.
 * C has not merged to "next" at all.
```

```
We would want to allow C to be rebased, refuse A, and encourage
B to be deleted.
To compute (1):
        git rev-list ^master ^topic next
        git rev-list ^master        next
        if these match, topic has not merged in next at all.
To compute (2):
        git rev-list master..topic
        if this is empty, it is fully merged to "master".
DOC_END
```

```
#!/bin/sh
#
# An example hook script to make use of push options.
# The example simply echoes all push options that start with 'echoback='
# and rejects all pushes when the "reject" push option is used.
#
# To enable this hook, rename this file to "pre-receive".
if test -n "$GIT_PUSH_OPTION_COUNT"
then
        i=0
        while test "$i" -lt "$GIT_PUSH_OPTION_COUNT"
        do
                eval "value=\$GIT_PUSH_OPTION_$i"
                case "$value" in
                echoback=*)
                        echo "echo from the pre-receive-hook: ${value#*=}" >&2
                        ;;
                reject)
                        exit 1
                esac
                i=$((i + 1))
        done
fi
```

```
#!/bin/sh
#
# An example hook script to prepare the commit log message.
# Called by "git commit" with the name of the file that has the
# commit message, followed by the description of the commit
# message's source.  The hook's purpose is to edit the commit
# message file.  If the hook fails with a non-zero status,
# the commit is aborted.
#
# To enable this hook, rename this file to "prepare-commit-msg".
# This hook includes three examples. The first one removes the
# "# Please enter the commit message..." help message.
#
# The second includes the output of "git diff --name-status -r"
# into the message, just before the "git status" output.  It is
# commented because it doesn't cope with --amend or with squashed
# commits.
#
# The third example adds a Signed-off-by line to the message, that can
# still be edited.  This is rarely a good idea.
COMMIT_MSG_FILE=$1
COMMIT_SOURCE=$2
SHA1=$3
/usr/bin/perl -i.bak -ne 'print unless(m/^. Please enter the commit
message/..m/^#$/)' "$COMMIT_MSG_FILE"
# case "$COMMIT_SOURCE,$SHA1" in
#  ,|template,)
#    /usr/bin/perl -i.bak -pe '
#       print "\n" . `git diff --cached --name-status -r`
#        if /^#/ && $first++ == 0' "$COMMIT_MSG_FILE" ;;
#  *) ;;
# esac
# SOB=$(git var GIT_COMMITTER_IDENT | sed -n 's/^\(.*>\).*$/Signed-off-by:
\1/p')
# git interpret-trailers --in-place --trailer "$SOB" "$COMMIT_MSG_FILE"
# if test -z "$COMMIT_SOURCE"
# then
#   /usr/bin/perl -i.bak -pe 'print "\n" if !$first_line++' "$COMMIT_MSG_FILE"
# fi
```

```
#!/bin/sh
#
# An example hook script to block unannotated tags from entering.
# Called by "git receive-pack" with arguments: refname sha1-old sha1-new
#
# To enable this hook, rename this file to "update".
#
# Config
# ------
# hooks.allowunannotated
#   This boolean sets whether unannotated tags will be allowed into the
#   repository.  By default they won't be.
# hooks.allowdeletetag
#   This boolean sets whether deleting tags will be allowed in the
#   repository.  By default they won't be.
# hooks.allowmodifytag
#   This boolean sets whether a tag may be modified after creation. By default
#   it won't be.
# hooks.allowdeletebranch
#   This boolean sets whether deleting branches will be allowed in the
#   repository.  By default they won't be.
# hooks.denycreatebranch
#   This boolean sets whether remotely creating branches will be denied
#   in the repository.  By default this is allowed.
#
# --- Command line
refname="$1"
oldrev="$2"
newrev="$3"
# --- Safety check
if [ -z "$GIT_DIR" ]; then
        echo "Don't run this script from the command line." >&2
        echo " (if you want, you could supply GIT_DIR then run" >&2
        echo "  $0 <ref> <oldrev> <newrev>)" >&2
        exit 1
fi
if [ -z "$refname" -o -z "$oldrev" -o -z "$newrev" ]; then
        echo "usage: $0 <ref> <oldrev> <newrev>" >&2
        exit 1
fi
# --- Config
allowunannotated=$(git config --bool hooks.allowunannotated)
allowdeletebranch=$(git config --bool hooks.allowdeletebranch)
```

```
denycreatebranch=$(git config --bool hooks.denycreatebranch)
allowdeletetag=$(git config --bool hooks.allowdeletetag)
allowmodifytag=$(git config --bool hooks.allowmodifytag)
# check for no description
projectdesc=$(sed -e '1q' "$GIT_DIR/description")
case "$projectdesc" in
"Unnamed repository"* | "")
        echo "*** Project description file hasn't been set" >&2
        exit 1
        ;;
esac
# --- Check types
# if $newrev is 0000...0000, it's a commit to delete a ref.
zero="0000000000000000000000000000000000000000"
if [ "$newrev" = "$zero" ]; then
        newrev_type=delete
else
        newrev_type=$(git cat-file -t $newrev)
fi
case "$refname","$newrev_type" in
        refs/tags/*,commit)
                # un-annotated tag
                short_refname=${refname##refs/tags/}
                if [ "$allowunannotated" != "true" ]; then
                        echo "*** The un-annotated tag, $short_refname, is not
allowed in this repository" >&2
                        echo "*** Use 'git tag [ -a | -s ]' for tags you want to
propagate." >&2
                        exit 1
                fi
                ;;
        refs/tags/*,delete)
                # delete tag
                if [ "$allowdeletetag" != "true" ]; then
                        echo "*** Deleting a tag is not allowed in this
repository" >&2
                        exit 1
                fi
                ;;
        refs/tags/*,tag)
                # annotated tag
                if [ "$allowmodifytag" != "true" ] && git rev-parse $refname >
/dev/null 2>&1
```

```
                then
                        echo "*** Tag '$refname' already exists." >&2
                        echo "*** Modifying a tag is not allowed in this
repository." >&2
                        exit 1
                fi
                ;;
        refs/heads/*,commit)
                # branch
                if [ "$oldrev" = "$zero" -a "$denycreatebranch" = "true" ]; then
                        echo "*** Creating a branch is not allowed in this
repository" >&2
                        exit 1
                fi
                ;;
        refs/heads/*,delete)
                # delete branch
                if [ "$allowdeletebranch" != "true" ]; then
                        echo "*** Deleting a branch is not allowed in this
repository" >&2
                        exit 1
                fi
                ;;
        refs/remotes/*,commit)
                # tracking branch
                ;;
        refs/remotes/*,delete)
                # delete tracking branch
                if [ "$allowdeletebranch" != "true" ]; then
                        echo "*** Deleting a tracking branch is not allowed in
this repository" >&2
                        exit 1
                fi
                ;;
        *)
                # Anything else (is there anything else?)
                echo "*** Update hook: unknown type of update to ref $refname of
type $newrev_type" >&2
                exit 1
                ;;
esac
# --- Finished
exit 0
```

```
<?xml version="1.0" encoding="utf-8"?>
<GameData>
  <LocalizedText>
   <!-- GERMAN -->
   <!-- Common Text -->
     <Replace Tag="LOC_TRADE_GROUP_SETTINGS" Language="de_DE">
        <Text>Gruppeneinstellungen</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_SETTINGS" Language="de_DE">
        <Text>Filtereinstellungen</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FOOD_TOOLTIP" Language="de_DE">
        <Text>Sort. n. [ICON_Food] Nahrung</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_PRODUCTION_TOOLTIP" Language="de_DE">
        <Text>Sort. n. [ICON_Production] Produktion</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_GOLD_TOOLTIP" Language="de_DE">
        <Text>Sort. n. [ICON_Gold] Gold</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_SCIENCE_TOOLTIP" Language="de_DE">
        <Text>Sort. n. [ICON_Science] Forschung</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_CULTURE_TOOLTIP" Language="de_DE">
        <Text>Sort.n. [ICON_Culture] Kultur</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FAITH_TOOLTIP" Language="de_DE">
        <Text>Sort. n. [ICON_Faith] Glaube</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TURNS_REMAINING_TOOLTIP" Language="de_DE">
        <Text>Sort. n. [ICON_Turn] Runden bis Route fertig ist</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_INTERNATIONAL_ROUTES_TEXT" Language="de_DE">
        <Text>Internationale Routen</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_CS_WITH_QUEST_TOOLTIP" Language="de_DE">
        <Text>Stadtstaaten mit Handelsquest</Text>
    </Replace>
    <!-- Trade Overview -->
    <Replace Tag="LOC_TRADE_NO_BENEFIT_TEXT" Language="de_DE">
        <Text>erhält keine Bonusse durch Handelsroute</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TEXT" Language="de_DE">
```

```
        <Text>Sort. nach</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_EXPAND_ALL_BUTTON_TEXT" Language="de_DE">
        <Text>Max</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_COLLAPSE_ALL_BUTTON_TEXT" Language="de_DE">
        <Text>Min</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_HELP_TOOLTIP" Language="de_DE">
        <Text>Gesamtbetrag an[ICON_Turn] bis zur Erstellung der
Handelsroute</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ALT_HELP_TOOLTIP" Language="de_DE">
        <Text>Diese Route benötigt {1_TurnsRemaining}[Icon_Turn] bis zur
Fertigstellung.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TOOLTIP_BREAKER" Language="de_DE">
        <Text>----------------------------</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ROUTE_LENGTH_TOOLTIP"
Language="de_DE">
        <Text>Handelsroute[ICON_Movement]: {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TRIPS_COUNT_TOOLTIP"
Language="de_DE">
        <Text>Züge bis zum Ziel: {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_TOOLTIP"
Language="de_DE">
        <Text>Route wird fertiggestellt in
[ICON_Turn]{1_TurnWhenCompleted}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_ALT_TOOLTIP"
Language="de_DE">
        <Text>Die begonnene Route benötigt {1_TurnsToComplete} Runden und wird
fertiggestellt in Runde {2_TurnWhenCompleted}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_CANCEL_AUTOMATION_TOOLTIP" Language="de_DE">
        <Text>Abbruch der automatisierten Handelsroute. Der Händler wird
verfügbar sein, sobald seine Route abgeschlossen ist.</Text>
    </Replace>
    <!-- Make Trade Route -->
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_LABEL" Language="de_DE">
```

```
        <Text>Wiederhole Route:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_TOOLTIP" Language="de_DE">
        <Text>Aktiviere, damit der Händler diese Route wiederholt. Standardmäßig
ist die wiederholte Route die letzte Route des Händlers.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_LABEL"
Language="de_DE">
        <Text>Von oben sort.Einträge:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_TOOLTIP"
Language="de_DE">
        <Text>Die wiederholte Route ist die oberste Route in der Sortierliste.
Die Route wird gewählt, sobald der Händler die Vorige abgeschlossen hat.</Text>
    </Replace>
    <!-- Trade Origin Chooser -->
    <Replace Tag="LOC_ORIGIN_CHOOSER_HEADER_BACKGROUND_TEXT" Language="de_DE">
        <Text>Wähle eine Stadt</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_ORIGIN_CHOOSER_HEADER_LABEL_TEXT" Language="de_DE">
        <Text>Transferiere den Händler nach...</Text>
    </Replace>
    <Replace Tag="LOC_ORIGIN_CHOOSER_CHANGE_DESTINATION_BUTTON"
Language="de_DE">
        <Text>Bestätige</Text>
    </Replace>
  </LocalizedText>
</GameData>
```

```
<?xml version="1.0" encoding="utf-8"?>
<GameData>
  <BaseGameText>
    <!-- ENGLISH -->
    <!-- Common Text -->
    <Row Tag="LOC_TRADE_GROUP_SETTINGS">
      <Text>Group Settings</Text>
    </Row>
    <Row Tag="LOC_TRADE_FILTER_SETTINGS">
      <Text>Filter Settings</Text>
    </Row>
    <Row Tag="LOC_TRADE_SORT_BY_FOOD_TOOLTIP">
      <Text>Sort by [ICON_Food]Food.</Text>
    </Row>
    <Row Tag="LOC_TRADE_SORT_BY_PRODUCTION_TOOLTIP">
      <Text>Sort by [ICON_Production]Production.</Text>
    </Row>
    <Row Tag="LOC_TRADE_SORT_BY_GOLD_TOOLTIP">
      <Text>Sort by [ICON_Gold]Gold.</Text>
    </Row>
    <Row Tag="LOC_TRADE_SORT_BY_SCIENCE_TOOLTIP">
      <Text>Sort by [ICON_Science]Science.</Text>
    </Row>
    <Row Tag="LOC_TRADE_SORT_BY_CULTURE_TOOLTIP">
      <Text>Sort by [ICON_Culture]Culture.</Text>
    </Row>
    <Row Tag="LOC_TRADE_SORT_BY_FAITH_TOOLTIP">
      <Text>Sort by [ICON_Faith]Faith.</Text>
    </Row>
    <Row Tag="LOC_TRADE_SORT_BY_TURNS_REMAINING_TOOLTIP">
      <Text>Sort by [ICON_Turn]Turns to complete route.</Text>
    </Row>
    <Row Tag="LOC_TRADE_FILTER_INTERNATIONAL_ROUTES_TEXT">
      <Text>International Routes</Text>
    </Row>
    <Row Tag="LOC_TRADE_FILTER_CS_WITH_QUEST_TOOLTIP">
      <Text>City-States with Trade Quest</Text>
    </Row>
    <!-- Trade Overview -->
    <Row Tag="LOC_TRADE_NO_BENEFIT_TEXT">
      <Text>gains no benefits from this Route.</Text>
    </Row>
    <Row Tag="LOC_TRADE_SORT_BY_TEXT">
```

```
      <Text>Sort by:</Text>
    </Row>
    <Row Tag="LOC_TRADE_OVERVIEW_ORIGIN_AZ">
      <Text>Origin A-Z</Text>
    </Row>
    <Row Tag="LOC_TRADE_OVERVIEW_ORIGIN_ZA">
      <Text>Origin Z-A</Text>
    </Row>
    <Row Tag="LOC_TRADE_OVERVIEW_DESTINATION_AZ">
      <Text>Destination A-Z</Text>
    </Row>
    <Row Tag="LOC_TRADE_OVERVIEW_DESTINATION_ZA">
      <Text>Destination Z-A</Text>
    </Row>
    <Row Tag="LOC_TRADE_EXPAND_ALL_BUTTON_TEXT">
      <Text>Exp:</Text>
    </Row>
    <Row Tag="LOC_TRADE_COLLAPSE_ALL_BUTTON_TEXT">
      <Text>Col:</Text>
    </Row>
    <Row Tag="LOC_TRADE_TURNS_REMAINING_HELP_TOOLTIP">
      <Text>Total amount of[ICON_Turn]to complete this trade route</Text>
    </Row>
    <Row Tag="LOC_TRADE_TURNS_REMAINING_ALT_HELP_TOOLTIP">
      <Text>This route will take {1_TurnsRemaining}[Icon_Turn] to
complete.</Text>
    </Row>
    <Row Tag="LOC_TRADE_TURNS_REMAINING_TOOLTIP_BREAKER">
      <Text>----------------------------</Text>
    </Row>
    <Row Tag="LOC_TRADE_TURNS_REMAINING_ROUTE_LENGTH_TOOLTIP">
      <Text>Trade Route[ICON_Movement]: {1_TripsToDestination}</Text>
    </Row>
    <Row Tag="LOC_TRADE_TURNS_REMAINING_TRIPS_COUNT_TOOLTIP">
      <Text>Trips to destination: {1_TripsToDestination}</Text>
    </Row>
    <Row Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_TOOLTIP">
      <Text>Route will complete in [ICON_Turn]{1_TurnWhenCompleted}</Text>
    </Row>
    <Row Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_ALT_TOOLTIP">
      <Text>If started, route will take {1_TurnsToComplete} turns and will
complete on turn {2_TurnWhenCompleted}</Text>
    </Row>
```

```
    <Row Tag="LOC_TRADE_CANCEL_AUTOMATION_TOOLTIP">
      <Text>Cancel this trade route's automation. The trader will be available,
once this route completes.</Text>
    </Row>
    <!-- Make Trade Route -->
    <Row Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_LABEL">
      <Text>Repeat Route:</Text>
    </Row>
    <Row Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_TOOLTIP">
      <Text>Enable, to have this trader repeat its route. By default, the route
repeated is the last one the trader made.</Text>
    </Row>
    <Row Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_LABEL">
      <Text>From Top Sort Entry:</Text>
    </Row>
    <Row Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_TOOLTIP">
      <Text>The route repeated is the top route from the currently sort
settings. The route gets picked when trader completes his last one.</Text>
    </Row>
    <!-- Trade Origin Chooser -->
    <Row Tag="LOC_ORIGIN_CHOOSER_HEADER_BACKGROUND_TEXT">
      <Text>Choose a city</Text>
    </Row>
    <Row Tag="LOC_TRADE_ORIGIN_CHOOSER_HEADER_LABEL_TEXT">
      <Text>TRANSFER TRADER TO...</Text>
    </Row>
    <Row Tag="LOC_ORIGIN_CHOOSER_CHANGE_DESTINATION_BUTTON">
      <Text>CONFIRM</Text>
    </Row>
  </BaseGameText>
</GameData>
```

```
<?xml version="1.0" encoding="utf-8"?>
<GameData>
  <LocalizedText>
    <!-- SPANISH -->
    <!-- Common Text -->
    <Replace Tag="LOC_TRADE_GROUP_SETTINGS" Language="es_ES">
      <Text>Grupo</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_SETTINGS" Language="es_ES">
      <Text>Filtro</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FOOD_TOOLTIP" Language="es_ES">
      <Text>Ordenar por [ICON_Food]Alimentos.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_PRODUCTION_TOOLTIP" Language="es_ES">
      <Text>Ordenar por [ICON_Production]Producción.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_GOLD_TOOLTIP" Language="es_ES">
      <Text>Ordenar por [ICON_Gold]Oro.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_SCIENCE_TOOLTIP" Language="es_ES">
      <Text>Ordenar por [ICON_Science]Ciencia.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_CULTURE_TOOLTIP" Language="es_ES">
      <Text>Ordenar por [ICON_Culture]Cultura.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FAITH_TOOLTIP" Language="es_ES">
      <Text>Ordenar por [ICON_Faith]Fe.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TURNS_REMAINING_TOOLTIP" Language="es_ES">
      <Text>Ordenar por [ICON_Turn]Turnos para completar la ruta.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_INTERNATIONAL_ROUTES_TEXT" Language="es_ES">
      <Text>Rutas Internacionales</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_CS_WITH_QUEST_TOOLTIP" Language="es_ES">
      <Text>C-E con Misión Comercial</Text>
    </Replace>
    <!-- Trade Overview -->
    <Replace Tag="LOC_TRADE_NO_BENEFIT_TEXT" Language="es_ES">
      <Text>no obtiene beneficios por esta ruta.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TEXT" Language="es_ES">
```

```
      <Text>Ordenar por:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_EXPAND_ALL_BUTTON_TEXT" Language="es_ES">
      <Text>Exp:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_COLLAPSE_ALL_BUTTON_TEXT" Language="es_ES">
      <Text>Con:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_HELP_TOOLTIP" Language="es_ES">
      <Text>Número de[ICON_Turn]para completar esta ruta</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ALT_HELP_TOOLTIP" Language="es_ES">
      <Text>Esta ruta se completará en {1_TurnsRemaining}[Icon_Turn].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TOOLTIP_BREAKER" Language="es_ES">
      <Text>----------------------------</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ROUTE_LENGTH_TOOLTIP"
Language="es_ES">
      <Text>Ruta Comercial[ICON_Movement]: {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TRIPS_COUNT_TOOLTIP"
Language="es_ES">
      <Text>Viajes hacia el destino: {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_TOOLTIP"
Language="es_ES">
      <Text>La ruta se completará en [ICON_Turn]{1_TurnWhenCompleted}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_ALT_TOOLTIP"
Language="es_ES">
      <Text>Una vez iniciada, la ruta durará {1_TurnsToComplete} turnos y se
completará en el turno {2_TurnWhenCompleted}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_CANCEL_AUTOMATION_TOOLTIP" Language="es_ES">
      <Text>Cancela la automatización de esta ruta. El comerciante estará
disponible una vez que la ruta se complete.</Text>
    </Replace>
    <!-- Make Trade Route -->
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_LABEL" Language="es_ES">
      <Text>Repetir</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_TOOLTIP" Language="es_ES">
```

```
      <Text>Activa para que este comerciante repita la ruta. Por defecto, la
ruta repetida es la última efectuada por el comerciante.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_LABEL"
Language="es_ES">
      <Text>Mejor ruta</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_TOOLTIP"
Language="es_ES">
      <Text>La ruta repetida es la mejor ruta según los criterios de ordenación
seleccionados. La nueva mejor ruta se escoge cada vez que el comerciante
complete su ruta actual.</Text>
    </Replace>
    <!-- Trade Origin Chooser -->
    <Replace Tag="LOC_ORIGIN_CHOOSER_HEADER_BACKGROUND_TEXT" Language="es_ES">
      <Text>Escoge una ciudad</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_ORIGIN_CHOOSER_HEADER_LABEL_TEXT" Language="es_ES">
      <Text>TRANSFERIR COMERCIANTE A...</Text>
    </Replace>
    <Replace Tag="LOC_ORIGIN_CHOOSER_CHANGE_DESTINATION_BUTTON"
Language="es_ES">
      <Text>CONFIRMAR</Text>
    </Replace>
  </LocalizedText>
</GameData>
```

```
<?xml version="1.0" encoding="utf-8"?>
<GameData>
  <LocalizedText>
    <!-- FRENCH -->
    <!-- Common Text -->
    <Replace Tag="LOC_TRADE_GROUP_SETTINGS" Language="fr_FR">
      <Text>Paramètres de groupe</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_SETTINGS" Language="fr_FR">
      <Text>Paramètres de filtrage</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FOOD_TOOLTIP" Language="fr_FR">
      <Text>Trier par [ICON_Food]Nourriture.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_PRODUCTION_TOOLTIP" Language="fr_FR">
      <Text>Trier par [ICON_Production]Production.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_GOLD_TOOLTIP" Language="fr_FR">
      <Text>Trier par [ICON_Gold]Or.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_SCIENCE_TOOLTIP" Language="fr_FR">
      <Text>Trier par [ICON_Science]Science.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_CULTURE_TOOLTIP" Language="fr_FR">
      <Text>Trier par [ICON_Culture]Culture.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FAITH_TOOLTIP" Language="fr_FR">
      <Text>Trier par [ICON_Faith]Foi.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TURNS_REMAINING_TOOLTIP" Language="fr_FR">
      <Text>Trier par [ICON_Turn]Tours pour compléter la route.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_INTERNATIONAL_ROUTES_TEXT" Language="fr_FR">
      <Text>Routes internationales</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_CS_WITH_QUEST_TOOLTIP" Language="fr_FR">
      <Text>Cité-États avec une Quête Commerciale</Text>
    </Replace>
    <!-- Trade Overview -->
    <Replace Tag="LOC_TRADE_NO_BENEFIT_TEXT" Language="fr_FR">
      <Text>Aucun avantage bénéfique de cette Route.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TEXT" Language="fr_FR">
```

```
      <Text>Trier par :</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_EXPAND_ALL_BUTTON_TEXT" Language="fr_FR">
      <Text>Exp :</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_COLLAPSE_ALL_BUTTON_TEXT" Language="fr_FR">
      <Text>Col :</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_HELP_TOOLTIP" Language="fr_FR">
      <Text>Nombre total de[ICON_Turn]pour compléter cette route
commerciale.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ALT_HELP_TOOLTIP" Language="fr_FR">
      <Text>Cette route prendra {1_TurnsRemaining}[Icon_Turn] pour être
terminée.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TOOLTIP_BREAKER" Language="fr_FR">
      <Text>----------------------------</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ROUTE_LENGTH_TOOLTIP"
Language="fr_FR">
      <Text>Route commerciale[ICON_Movement] : {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TRIPS_COUNT_TOOLTIP"
Language="fr_FR">
      <Text>Voyages vers la destination : {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_TOOLTIP"
Language="fr_FR">
      <Text>La Route sera terminée dans [ICON_Turn]{1_TurnWhenCompleted}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_ALT_TOOLTIP"
Language="fr_FR">
      <Text>Si démarré, l'itinéraire prend {1_TurnsToComplete} tours et se
terminera au tour {2_TurnWhenCompleted}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_CANCEL_AUTOMATION_TOOLTIP" Language="fr_FR">
      <Text>Annulez l'automatisation de cette route commerciale. Le commerçant
sera disponible, une fois cette route complète.</Text>
    </Replace>
    <!-- Make Trade Route -->
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_LABEL" Language="fr_FR">
      <Text>Reproduire la route :</Text>
```

```
    </Replace>
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_TOOLTIP" Language="fr_FR">
      <Text>Activer, pour que le marchand reproduise sa route. Par défaut, la
route reproduite est la dernière que le marchand a faite.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_LABEL"
Language="fr_FR">
      <Text>De Haut Tri Entrée:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_TOOLTIP"
Language="fr_FR">
      <Text>L'itinéraire reproduit est l'itinéraire le plus haut des paramètres
de tri actuels. L'itinéraire est choisi lorsque le commerçant termine son
dernier.</Text>
    </Replace>
    <!-- Trade Origin Chooser -->
    <Replace Tag="LOC_ORIGIN_CHOOSER_HEADER_BACKGROUND_TEXT" Language="fr_FR">
      <Text>Choisissez une ville</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_ORIGIN_CHOOSER_HEADER_LABEL_TEXT" Language="fr_FR">
      <Text>TRANSFÉRER LE MARCHAND VERS...</Text>
    </Replace>
    <Replace Tag="LOC_ORIGIN_CHOOSER_CHANGE_DESTINATION_BUTTON"
Language="fr_FR">
      <Text>CONFIRMER</Text>
    </Replace>
  </LocalizedText>
</GameData>
```

```
<?xml version="1.0" encoding="utf-8"?>
<GameData>
  <LocalizedText>
    <!-- SIMPLIFY-CHINESE -->
    <!-- Common Text -->
    <Replace Tag="LOC_TRADE_GROUP_SETTINGS" Language="zh_Hans_CN">
      <Text>■■■■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_SETTINGS" Language="zh_Hans_CN">
      <Text>■■■■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FOOD_TOOLTIP" Language="zh_Hans_CN">
      <Text>■ [ICON_Food] ■■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_PRODUCTION_TOOLTIP" Language="zh_Hans_CN">
      <Text>■ [ICON_Production] ■■■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_GOLD_TOOLTIP" Language="zh_Hans_CN">
      <Text>■ [ICON_Gold] ■■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_SCIENCE_TOOLTIP" Language="zh_Hans_CN">
      <Text>■ [ICON_Science] ■■■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_CULTURE_TOOLTIP" Language="zh_Hans_CN">
      <Text>■ [ICON_Culture] ■■■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FAITH_TOOLTIP" Language="zh_Hans_CN">
      <Text>■ [ICON_Faith] ■■■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TURNS_REMAINING_TOOLTIP"
Language="zh_Hans_CN">
      <Text>■ [ICON_Turn] ■■■■■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_INTERNATIONAL_ROUTES_TEXT"
Language="zh_Hans_CN">
      <Text>■■■■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_CS_WITH_QUEST_TOOLTIP" Language="zh_Hans_CN">
      <Text>■■■■■■■■■■</Text>
    </Replace>
    <!-- Trade Overview -->
    <Replace Tag="LOC_TRADE_NO_BENEFIT_TEXT" Language="zh_Hans_CN">
      <Text>■■■■■■■■■■■■</Text>
```

```
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TEXT" Language="zh_Hans_CN">
      <Text>■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_EXPAND_ALL_BUTTON_TEXT" Language="zh_Hans_CN">
      <Text>■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_COLLAPSE_ALL_BUTTON_TEXT" Language="zh_Hans_CN">
      <Text>■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_HELP_TOOLTIP" Language="zh_Hans_CN">
      <Text>■■■■■■ [ICON_Turn] ■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ALT_HELP_TOOLTIP"
Language="zh_Hans_CN">
      <Text>■■■■■■■ {1_TurnsRemaining}[Icon_Turn] ■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ALT2_HELP_TOOLTIP"
Language="zh_Hans_CN">
      <Text>■■■■■■■ {1_TurnsRemaining}[Icon_Turn] ■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TOOLTIP_BREAKER"
Language="zh_Hans_CN">
      <Text>--------------------------</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ROUTE_LENGTH_TOOLTIP"
Language="zh_Hans_CN">
      <Text>■■■■ [ICON_Movement]: {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TRIPS_COUNT_TOOLTIP"
Language="zh_Hans_CN">
      <Text>■■■■: {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_TOOLTIP"
Language="zh_Hans_CN">
      <Text>■■■■■■ [ICON_Turn]{1_TurnWhenCompleted} ■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_ALT_TOOLTIP"
Language="zh_Hans_CN">
      <Text>■■■■■■■■■■■■■ [ICON_Turn]{1_TurnWhenCompleted} ■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_CANCEL_AUTOMATION_TOOLTIP" Language="zh_Hans_CN">
      <Text>■■■■■■■■■■■■■■■■■■■■■■■</Text>
```

```
    </Replace>
    <!-- Make Trade Route -->
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_LABEL" Language="zh_Hans_CN">
      <Text>[redacted]</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_TOOLTIP"
Language="zh_Hans_CN">
      <Text>[redacted]</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_LABEL"
Language="zh_Hans_CN">
      <Text>[redacted]</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_TOOLTIP"
Language="zh_Hans_CN">
      <Text>[redacted]</Text>
    </Replace>
    <!-- Trade Origin Chooser -->
    <Replace Tag="LOC_ORIGIN_CHOOSER_HEADER_BACKGROUND_TEXT"
Language="zh_Hans_CN">
      <Text>[redacted]</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_ORIGIN_CHOOSER_HEADER_LABEL_TEXT"
Language="zh_Hans_CN">
      <Text>[redacted]</Text>
    </Replace>
    <Replace Tag="LOC_ORIGIN_CHOOSER_CHANGE_DESTINATION_BUTTON"
Language="zh_Hans_CN">
      <Text>[redacted]</Text>
    </Replace>
  </LocalizedText>
</GameData>
```

```
<?xml version="1.0" encoding="utf-8"?>
<GameData>
  <LocalizedText>
    <!-- ITALIAN -->
    <!-- Common Text -->
    <Replace Tag="LOC_TRADE_GROUP_SETTINGS" Language="it_IT">
        <Text>Impostazioni gruppo</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_SETTINGS" Language="it_IT">
        <Text>Impostazioni filtro</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FOOD_TOOLTIP" Language="it_IT">
      <Text>Ordina per [ICON_Food]Cibo.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_PRODUCTION_TOOLTIP" Language="it_IT">
      <Text>Ordina per [ICON_Production]Produzione.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_GOLD_TOOLTIP" Language="it_IT">
      <Text>Ordina per [ICON_Gold]Oro.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_SCIENCE_TOOLTIP" Language="it_IT">
      <Text>Ordina per [ICON_Science]Scienza.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_CULTURE_TOOLTIP" Language="it_IT">
      <Text>Ordina per [ICON_Culture]Cultura.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FAITH_TOOLTIP" Language="it_IT">
      <Text>Ordina per [ICON_Faith]Fede.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TURNS_REMAINING_TOOLTIP" Language="it_IT">
      <Text>Ordina per [ICON_Turn]Turni al completamento della rotta.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_INTERNATIONAL_ROUTES_TEXT" Language="it_IT">
      <Text>Rotte Internazionali</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_CS_WITH_QUEST_TOOLTIP" Language="it_IT">
      <Text>Città Stato con Missione Rotta Commerciale</Text>
    </Replace>
    <!-- Trade Overview -->
    <Replace Tag="LOC_TRADE_NO_BENEFIT_TEXT" Language="it_IT">
      <Text>non ha benefici da questa Rotta Commerciale.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TEXT" Language="it_IT">
```

```
      <Text>Ordina per:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_EXPAND_ALL_BUTTON_TEXT" Language="it_IT">
      <Text>Esp:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_COLLAPSE_ALL_BUTTON_TEXT" Language="it_IT">
      <Text>Com:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_HELP_TOOLTIP" Language="it_IT">
      <Text>Numero di[ICON_Turn]al termine della rotta commerciale</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ALT_HELP_TOOLTIP" Language="it_IT">
      <Text>Questa rotta verrà completata in
{1_TurnsRemaining}[Icon_Turn].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TOOLTIP_BREAKER" Language="it_IT">
      <Text>----------------------------</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ROUTE_LENGTH_TOOLTIP"
Language="it_IT">
      <Text>Rotta Commerciale[ICON_Movement]: {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TRIPS_COUNT_TOOLTIP"
Language="it_IT">
      <Text>Viaggi a destinazione: {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_TOOLTIP"
Language="it_IT">
      <Text>La rotta verrà completata in [ICON_Turn]{1_TurnWhenCompleted}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_ALT_TOOLTIP"
Language="it_IT">
      <Text>Se avviata, la rotta verrà completata in {1_TurnsToComplete} turni e
si concluderà al turno {2_TurnWhenCompleted}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_CANCEL_AUTOMATION_TOOLTIP" Language="it_IT">
      <Text>Cancella l'automazione di questa rotta commerciale. Il commerciante
sarà disponibile al suo completamento.</Text>
    </Replace>
    <!-- Make Trade Route -->
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_LABEL" Language="it_IT">
      <Text>Ripeti Rotta:</Text>
    </Replace>
```

```
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_TOOLTIP" Language="it_IT">
      <Text>Abilita per ripetere la rotta. Di default, la rotta commerciale
ripetuta è l'ultima portata a termine dal commerciante.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_LABEL"
Language="it_IT">
      <Text>Ordina dall'alto per:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_TOOLTIP"
Language="it_IT">
      <Text>La rotta ripetuta è la migliore secondo l'attuale criterio di
ordinamento. La rotta verrà scelta quando il commerciante terminerà quella
corrente.</Text>
    </Replace>
    <!-- Trade Origin Chooser -->
    <Replace Tag="LOC_ORIGIN_CHOOSER_HEADER_BACKGROUND_TEXT" Language="it_IT">
      <Text>Scegli una città</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_ORIGIN_CHOOSER_HEADER_LABEL_TEXT" Language="it_IT">
      <Text>TRASFERISCI COMMERCIANTE A...</Text>
    </Replace>
    <Replace Tag="LOC_ORIGIN_CHOOSER_CHANGE_DESTINATION_BUTTON"
Language="it_IT">
      <Text>CONFERMA</Text>
    </Replace>
  </LocalizedText>
</GameData>
```

```
<?xml version="1.0" encoding="utf-8"?>
<GameData>
  <LocalizedText>
    <!-- KOREAN -->
    <!-- Common Text -->
    <Replace Tag="LOC_TRADE_GROUP_SETTINGS" Language="ko_KR">
      <Text>■■ ■■■ ■■■■■.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_SETTINGS" Language="ko_KR">
      <Text>■■ ■■■ ■■■■■.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FOOD_TOOLTIP" Language="ko_KR">
      <Text>[ICON_Food]■■ ■■■ ■■■■■.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_PRODUCTION_TOOLTIP" Language="ko_KR">
      <Text>[ICON_Production]■■■ ■■■ ■■■■■.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_GOLD_TOOLTIP" Language="ko_KR">
      <Text>[ICON_Gold]■ ■■■ ■■■■■.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_SCIENCE_TOOLTIP" Language="ko_KR">
      <Text>[ICON_Science]■■ ■■■ ■■■■■.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_CULTURE_TOOLTIP" Language="ko_KR">
      <Text>[ICON_Culture]■■ ■■■ ■■■■■.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FAITH_TOOLTIP" Language="ko_KR">
      <Text>[ICON_Faith]■■ ■■■ ■■■■■.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TURNS_REMAINING_TOOLTIP" Language="ko_KR">
      <Text>■■■ ■■■ [ICON_Turn]■ ■■ ■■■■■.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_INTERNATIONAL_ROUTES_TEXT" Language="ko_KR">
      <Text>■■ ■■■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_CS_WITH_QUEST_TOOLTIP" Language="ko_KR">
      <Text>■■■ ■■■ ■■■ ■■ ■■</Text>
    </Replace>
    <!-- Trade Overview -->
    <Replace Tag="LOC_TRADE_NO_BENEFIT_TEXT" Language="ko_KR">
      <Text>■ ■■■■ ■■ ■■ ■■■ ■■■■.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TEXT" Language="ko_KR">
```

```
      <Text>■■:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_EXPAND_ALL_BUTTON_TEXT" Language="ko_KR">
      <Text>■■:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_COLLAPSE_ALL_BUTTON_TEXT" Language="ko_KR">
      <Text>■■:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_HELP_TOOLTIP" Language="ko_KR">
      <Text>■ ■■■■ ■■■ [ICON_Turn]■ ■</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ALT_HELP_TOOLTIP" Language="ko_KR">
      <Text>■ ■■■■ [Icon_Turn]{1_TurnsRemaining}■ ■■ ■■■■■.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ALT2_HELP_TOOLTIP" Language="ko_KR">
      <Text>■ ■■■■ [Icon_Turn]{1_TurnsRemaining}■ ■■ ■■■■■.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TOOLTIP_BREAKER" Language="ko_KR">
      <Text>----------------------------</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ROUTE_LENGTH_TOOLTIP"
Language="ko_KR">
      <Text>■■■ ■■ ■■: [ICON_Movement]{1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TRIPS_COUNT_TOOLTIP"
Language="ko_KR">
      <Text>■■■■■■ ■■: {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_TOOLTIP"
Language="ko_KR">
      <Text>■■■■ [ICON_Turn]{1_TurnWhenCompleted}■■■ ■■■■■.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_ALT_TOOLTIP"
Language="ko_KR">
      <Text>■■ ■■■■ [ICON_Turn]{1_TurnsToComplete}■■ ■■■
[ICON_Turn]{2_TurnWhenCompleted}■■■ ■■■■■.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_CANCEL_AUTOMATION_TOOLTIP" Language="ko_KR">
      <Text>■ ■■■■ ■■ ■■■ ■■■. ■■■ ■■■ ■■■ ■■ ■■■ ■ ■■■■.</Text>
    </Replace>
    <!-- Make Trade Route -->
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_LABEL" Language="ko_KR">
      <Text>■■ ■■ ■■■:</Text>
```

```
    </Replace>
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_TOOLTIP" Language="ko_KR">
      <Text>[illegible].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_LABEL"
Language="ko_KR">
      <Text>[illegible]:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_TOOLTIP"
Language="ko_KR">
      <Text>[illegible].</Text>
    </Replace>
    <!-- Trade Origin Chooser -->
    <Replace Tag="LOC_ORIGIN_CHOOSER_HEADER_BACKGROUND_TEXT" Language="ko_KR">
      <Text>[illegible]</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_ORIGIN_CHOOSER_HEADER_LABEL_TEXT" Language="ko_KR">
      <Text>[illegible]</Text>
    </Replace>
    <Replace Tag="LOC_ORIGIN_CHOOSER_CHANGE_DESTINATION_BUTTON"
Language="ko_KR">
      <Text>[illegible]</Text>
    </Replace>
  </LocalizedText>
</GameData>
```

```
<?xml version="1.0" encoding="utf-8"?>
<GameData>
  <LocalizedText>
    <!-- POLISH -->
    <!-- Common Text -->
    <Replace Tag="LOC_TRADE_GROUP_SETTINGS" Language="pl_PL">
        <Text>Ustawienia grup</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_SETTINGS" Language="pl_PL">
        <Text>Ustawienia filtrów</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FOOD_TOOLTIP" Language="pl_PL">
      <Text>Sortuj po [ICON_Food]Jedzeniu.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_PRODUCTION_TOOLTIP" Language="pl_PL">
      <Text>Sortuj po [ICON_Production]Produkcji.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_GOLD_TOOLTIP" Language="pl_PL">
      <Text>Sortuj po [ICON_Gold]Z■ocie.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_SCIENCE_TOOLTIP" Language="pl_PL">
      <Text>Sortuj po [ICON_Science]Nauce.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_CULTURE_TOOLTIP" Language="pl_PL">
      <Text>Sortuj po [ICON_Culture]Kulturze.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FAITH_TOOLTIP" Language="pl_PL">
      <Text>Sortuj po [ICON_Faith]Wierze.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TURNS_REMAINING_TOOLTIP" Language="pl_PL">
      <Text>Sortuj po [ICON_Turn]liczbie tur do zako■czenia szlaku.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_INTERNATIONAL_ROUTES_TEXT" Language="pl_PL">
      <Text>Mi■dzynarodowe szlaki</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_CS_WITH_QUEST_TOOLTIP" Language="pl_PL">
      <Text>Miasta-pa■stwa z zadaniami</Text>
    </Replace>
    <!-- Trade Overview -->
    <Replace Tag="LOC_TRADE_NO_BENEFIT_TEXT" Language="pl_PL">
      <Text>nie zyskuje nic z tego szlaku.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TEXT" Language="pl_PL">
```

```
      <Text>Sortuj po:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_EXPAND_ALL_BUTTON_TEXT" Language="pl_PL">
        <Text>Rw:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_COLLAPSE_ALL_BUTTON_TEXT" Language="pl_PL">
        <Text>Zw:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_HELP_TOOLTIP" Language="pl_PL">
      <Text>Ca■kowita ilo■■ [ICON_Turn]do zako■czenia tego szlaku
handlowego</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ALT_HELP_TOOLTIP" Language="pl_PL">
      <Text>Zako■czenie tego szlaku zajmie {1_TurnsRemaining}[Icon_Turn].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TOOLTIP_BREAKER" Language="pl_PL">
      <Text>----------------------------</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ROUTE_LENGTH_TOOLTIP"
Language="pl_PL">
      <Text>Szlak handlowy[ICON_Movement]: {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TRIPS_COUNT_TOOLTIP"
Language="pl_PL">
      <Text>Podró■e do celu: {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_TOOLTIP"
Language="pl_PL">
      <Text>Szlak zako■czy si■ w turze: [ICON_Turn]{1_TurnWhenCompleted}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_ALT_TOOLTIP"
Language="pl_PL">
      <Text>Je■li rozpocz■ty, szlak te zajmie {1_TurnsToComplete} turn i
zostanie zako■czony w turze {2_TurnWhenCompleted}.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_CANCEL_AUTOMATION_TOOLTIP" Language="pl_PL">
      <Text>Wy■■cz automatyzacj■ tego szlaku handlowego. Kupiec b■dzie dost■pny
po zako■czeniu szlaku.</Text>
    </Replace>
    <!-- Make Trade Route -->
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_LABEL" Language="pl_PL">
      <Text>Powtarzaj:</Text>
    </Replace>
```

```
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_TOOLTIP" Language="pl_PL">
      <Text>W■■cz by ten kupiec powtarza■ swój szlak. Standardowo, powtórzony
zostanie ostatni szlak wykonany przez kupca.</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_LABEL"
Language="pl_PL">
      <Text>Powtarzaj wed■ug sortowania:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_TOOLTIP"
Language="pl_PL">
      <Text>Powtórzony zostanie szlak z góry aktualnie sortowanej listy. Szlak
zostanie wybrany gdy kupiec zako■czy ostatni szlak.</Text>
    </Replace>
    <!-- Trade Origin Chooser -->
    <Replace Tag="LOC_ORIGIN_CHOOSER_HEADER_BACKGROUND_TEXT" Language="pl_PL">
      <Text>Wybierz miasto</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_ORIGIN_CHOOSER_HEADER_LABEL_TEXT" Language="pl_PL">
      <Text>PRZENIE■ KUPCA DO...</Text>
    </Replace>
    <Replace Tag="LOC_ORIGIN_CHOOSER_CHANGE_DESTINATION_BUTTON"
Language="pl_PL">
      <Text>POTWIERD■</Text>
    </Replace>
  </LocalizedText>
</GameData>
```

```
<?xml version="1.0" encoding="utf-8"?>
<GameData>
  <LocalizedText>
    <!-- RUSSIAN -->
    <!-- Common Text -->
    <Replace Tag="LOC_TRADE_GROUP_SETTINGS" Language="ru_RU">
        <Text>[redacted]</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_SETTINGS" Language="ru_RU">
        <Text>[redacted]</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FOOD_TOOLTIP" Language="ru_RU">
      <Text>[redacted] [ICON_Food][redacted].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_PRODUCTION_TOOLTIP" Language="ru_RU">
      <Text>[redacted] [ICON_Production][redacted].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_GOLD_TOOLTIP" Language="ru_RU">
      <Text>[redacted] [ICON_Gold][redacted].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_SCIENCE_TOOLTIP" Language="ru_RU">
      <Text>[redacted] [ICON_Science][redacted].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_CULTURE_TOOLTIP" Language="ru_RU">
      <Text>[redacted] [ICON_Culture][redacted].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_FAITH_TOOLTIP" Language="ru_RU">
      <Text>[redacted] [ICON_Faith][redacted].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TURNS_REMAINING_TOOLTIP" Language="ru_RU">
      <Text>[redacted] [ICON_Turn][redacted] [redacted] [redacted].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_INTERNATIONAL_ROUTES_TEXT" Language="ru_RU">
      <Text>[redacted] [redacted]</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FILTER_CS_WITH_QUEST_TOOLTIP" Language="ru_RU">
      <Text>[redacted]-[redacted] [redacted] [redacted]</Text>
    </Replace>
    <!-- Trade Overview -->
    <Replace Tag="LOC_TRADE_NO_BENEFIT_TEXT" Language="ru_RU">
      <Text>[redacted] [redacted] [redacted] [redacted].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_SORT_BY_TEXT" Language="ru_RU">
```

```
      <Text>[redacted]:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_EXPAND_ALL_BUTTON_TEXT" Language="ru_RU">
      <Text>[redacted]:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_COLLAPSE_ALL_BUTTON_TEXT" Language="ru_RU">
      <Text>[redacted]:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_HELP_TOOLTIP" Language="ru_RU">
      <Text>[redacted] [ICON_Turn] [redacted]</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ALT_HELP_TOOLTIP" Language="ru_RU">
      <Text>[redacted] {1_TurnsRemaining}[Icon_Turn].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TOOLTIP_BREAKER" Language="ru_RU">
      <Text>----------</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_ROUTE_LENGTH_TOOLTIP"
Language="ru_RU">
      <Text>[redacted][ICON_Movement]: {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TRIPS_COUNT_TOOLTIP"
Language="ru_RU">
      <Text>[redacted]: {1_TripsToDestination}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_TOOLTIP"
Language="ru_RU">
      <Text>[redacted]: [ICON_Turn]{1_TurnWhenCompleted}</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_ALT_TOOLTIP"
Language="ru_RU">
      <Text>[redacted], [redacted] {1_TurnsToComplete} [redacted]
{2_TurnWhenCompleted} [redacted].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_CANCEL_AUTOMATION_TOOLTIP" Language="ru_RU">
      <Text>[redacted] [redacted]. [redacted] [redacted], [redacted].</Text>
    </Replace>
    <!-- Make Trade Route -->
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_LABEL" Language="ru_RU">
      <Text>[redacted]:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_TOOLTIP" Language="ru_RU">
```

```
      <Text>[illegible], [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible],
[illegible] [illegible] [illegible] [illegible].</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_LABEL"
Language="ru_RU">
      <Text>[illegible] [illegible] [illegible]:</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_TOOLTIP"
Language="ru_RU">
      <Text>[illegible] [illegible] – [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible]
[illegible] [illegible], [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].</Text>
    </Replace>
    <!-- Trade Origin Chooser -->
    <Replace Tag="LOC_ORIGIN_CHOOSER_HEADER_BACKGROUND_TEXT" Language="ru_RU">
      <Text>[illegible] [illegible]</Text>
    </Replace>
    <Replace Tag="LOC_TRADE_ORIGIN_CHOOSER_HEADER_LABEL_TEXT" Language="ru_RU">
      <Text>[illegible] [illegible] [illegible]...</Text>
    </Replace>
    <Replace Tag="LOC_ORIGIN_CHOOSER_CHANGE_DESTINATION_BUTTON"
Language="ru_RU">
      <Text>[illegible]</Text>
    </Replace>
  </LocalizedText>
</GameData>
```

```
-- ===========================================================================
--  SETTINGS
-- ===========================================================================
local alignTradeYields = true
local showNoBenefitsString = false
local showSortOrdersPermanently = false
local hideTradingPostIcon = false
-- Color Settings for Headers
local colorCityPlayerHeader = true
local backdropGridColorOffset = 20
local backdropGridColorOpacity = 140
local backdropColorOffset = -15
local backdropColorOpacity = 55
local labelColorOffset = -27
local labelColorOpacity = 255
-- Color Settings for Route Entry
local tintRouteEntry = false
local tintColorOffset = 80
local tintColorOpacity = 205
-- Set to false to hide all debug prints from this file.
local dbug_print = true
local opt_print = false
-- ===========================================================================
--  INCLUDES
-- ===========================================================================
include("AnimSidePanelSupport");
include("PopupDialogSupport");
include("InstanceManager");
include("SupportFunctions");
include("TradeSupport");
-- ===========================================================================
--  CONSTANTS
-- ===========================================================================
local RELOAD_CACHE_ID:string = "TradeOverview"; -- Must be unique (usually the
same as the file name)
local OUTSIDE_SUPPORT_CACHE_ID:string = "TradeOverviewSupport";
local DATA_ICON_PREFIX:string = "ICON_";
local TRADE_TABS:table = {
    MY_ROUTES           = 0;
    ROUTES_TO_CITIES    = 1;
    AVAILABLE_ROUTES    = 2;
};
local GROUP_BY_SETTINGS:table = {
```

```
    NONE                = 1;
    ORIGIN              = 2;
    DESTINATION         = 3;
    -- Special group by's (these get converted to sort settings in
OnGroupBySelected)
    ORIGIN_AZ           = 4;
    ORIGIN_ZA           = 5;
    DESTINATION_AZ      = 6;
    DESTINATION_ZA      = 7;
};
local SEMI_EXPAND_SETTINGS:table = {};
SEMI_EXPAND_SETTINGS[GROUP_BY_SETTINGS.ORIGIN] = 4;
SEMI_EXPAND_SETTINGS[GROUP_BY_SETTINGS.DESTINATION] = 2;
local BASE_TOURISM_MODIFIER = GlobalParameters.TOURISM_TRADE_ROUTE_BONUS;
-- ===========================================================================
--  VARIABLES
-- ===========================================================================
local m_RouteInstanceIM:table           = InstanceManager:new("RouteInstance",
"Top", Controls.BodyStack);
local m_HeaderInstanceIM:table          = InstanceManager:new("HeaderInstance",
"Top", Controls.BodyStack);
local m_SimpleButtonInstanceIM:table    =
InstanceManager:new("SimpleButtonInstance", "Top", Controls.BodyStack);
local m_DividerInstanceIM:table         =
InstanceManager:new("SectionDividerInstance", "Top", Controls.BodyStack);
local m_AnimSupport:table; -- AnimSidePanelSupport
local m_currentTab:number = TRADE_TABS.MY_ROUTES;
local m_shiftDown:boolean = false;
local m_ctrlDown:boolean = false;
local m_sortCallRefresh:boolean = false;
-- Trade Routes Tables
local m_AvailableTradeRoutes:table = {};        -- Stores all available routes
local m_FinalTradeRoutes:table = {};            -- Filter version of above
local m_AvailableGroupedRoutes:table = {};      -- Grouped version of routes.
Built from above
local m_AvailableTraders:table = {};            -- Indexed by the city id, value
stored is the unit id
local m_TurnUpdatedTraders:number = -1;
-- Stores filter list and tracks the currently selected list
local m_filterList:table = {};
local m_filterCount:number = 0;
local m_filterSelected:number = 1;
local m_groupBySelected:number = GROUP_BY_SETTINGS.DESTINATION;
```

```
local m_groupByList:table = {};
local m_GroupExpandAll:boolean = false;
local m_GroupCollapseAll:boolean = false;
local m_GroupsFullyExpanded:table = {};
local m_GroupsFullyCollapsed:table = {};
local m_HasBuiltTradeRouteTable:boolean = false;
local m_LastTurnBuiltTradeRouteTable:number = -1;
local m_SortSettingsChanged:boolean = true;
local m_GroupSettingsChanged:boolean = true;
local m_FilterSettingsChanged:boolean = true;
-- Stores the sort settings.
local m_InGroupSortBySettings = {}; -- Stores the setting each group will have
within it. Applicable when routes are grouped
local m_GroupSortBySettings = {}; -- Stores the overall group sort setting. This
is used, when routes are NOT grouped
local m_dividerCount = 0
-- ===========================================================================
--  Refresh functions
-- ===========================================================================
-- Finds and adds all possible trade routes
function RebuildAvailableTradeRoutesTable()
    if dbug_print then
        print("Rebuilding Trade Routes table");
    end
    m_AvailableTradeRoutes = {};
    local sourcePlayerID = Game.GetLocalPlayer();
    local sourceCities:table = Players[sourcePlayerID]:GetCities();
    local players:table = Game.GetPlayers{ Alive=true };
    local destinationCitiesID:table = {};
    local tradeManager:table = Game.GetTradeManager();
    for _, sourceCity in sourceCities:Members() do
        local sourceCityID:number = sourceCity:GetID();
        for _, destinationPlayer in ipairs(players) do
            local destinationPlayerID:number = destinationPlayer:GetID()
            -- Check for war, met, etc
            if CanPossiblyTradeWithPlayer(sourcePlayerID, destinationPlayerID)
then
                for _, destinationCity in
destinationPlayer:GetCities():Members() do
                    local destinationCityID:number = destinationCity:GetID();
                    if tradeManager:CanStartRoute(sourcePlayerID, sourceCityID,
destinationPlayerID, destinationCityID) then
                        -- Create the trade route entry
```

```
                        local tradeRoute = {
                            OriginCityPlayer        = sourcePlayerID,
                            OriginCityID            = sourceCityID,
                            DestinationCityPlayer   = destinationPlayerID,
                            DestinationCityID       = destinationCityID
                        };
                        table.insert(m_AvailableTradeRoutes, tradeRoute);
                    end
                end
            end
        end
    end
    if dbug_print then
        print("Total routes = " .. table.count(m_AvailableTradeRoutes))
    end
    m_HasBuiltTradeRouteTable = true;
    m_LastTurnBuiltTradeRouteTable = Game.GetCurrentGameTurn();
end
function RebuildAvailableTraders()
    if dbug_print then
        print("Building available traders")
    end
    local playerID = Game.GetLocalPlayer()
    local pPlayer = Players[playerID]
    local pPlayerUnits = pPlayer:GetUnits()
    m_AvailableTraders = {}
    for i, pUnit in pPlayerUnits:Members() do
        local unitInfo:table = GameInfo.Units[pUnit:GetUnitType()];
        local unitID:number = pUnit:GetID();
        if unitInfo.MakeTradeRoute == true and (not
pUnit:HasPendingOperations()) then
            local pCity = Cities.GetCityInPlot(pUnit:GetX(), pUnit:GetY());
            if pCity ~= nil then
                local cityID = pCity:GetID()
                -- Make entry if none exists
                if m_AvailableTraders[cityID] == nil then
                    m_AvailableTraders[cityID] = {}
                end
                -- Append unit into the entry
                table.insert(m_AvailableTraders[cityID], unitID)
            end
        end
    end
```

```
    m_TurnUpdatedTraders = Game.GetCurrentGameTurn()
end
function Refresh()
    local time1 = Automation.GetTime();
    if dbug_print then
        print("Refresh start")
    end
    PreRefresh();
    RefreshGroupByPulldown();
    RefreshFilters();
    RefreshSortBar();
    if m_TurnUpdatedTraders < Game.GetCurrentGameTurn() then
        RebuildAvailableTraders()
    end
    if m_currentTab == TRADE_TABS.MY_ROUTES then
        ViewMyRoutes();
    elseif m_currentTab == TRADE_TABS.ROUTES_TO_CITIES then
        ViewRoutesToCities();
    elseif m_currentTab == TRADE_TABS.AVAILABLE_ROUTES then
        ViewAvailableRoutes();
    else
        ViewMyRoutes();
    end
    PostRefresh();
    local time2 = Automation.GetTime()
    if dbug_print then
        print(string.format("Time taken to refresh: %.4f sec(s)", time2-time1))
    end
end
function PreRefresh()
    -- Reset Stack
    m_RouteInstanceIM:ResetInstances();
    m_HeaderInstanceIM:ResetInstances();
    m_SimpleButtonInstanceIM:ResetInstances();
    m_DividerInstanceIM:ResetInstances();
    m_dividerCount = 0
end
function PostRefresh()
    -- Calculate Stack Sizes
    local time1 = Automation.GetTime()
    Controls.HeaderStack:CalculateSize();
    Controls.HeaderStack:ReprocessAnchoring();
    Controls.BodyScrollPanel:CalculateSize();
```

```
    Controls.BodyScrollPanel:ReprocessAnchoring();
    Controls.BodyScrollPanel:CalculateInternalSize();
    local time2 = Automation.GetTime()
    if dbug_print then
        print(string.format("Time to calculate stack sizes: %.4f sec(s)",
time2-time1))
    end
end
-- ===========================================================================
--  Tab functions
-- ===========================================================================
-- Show My Routes Tab
function ViewMyRoutes()
    -- Update Tabs
    SetMyRoutesTabSelected(true);
    SetRoutesToCitiesTabSelected(false);
    SetAvailableRoutesTabSelected(false);
    local localPlayerID = Game.GetLocalPlayer();
    if (localPlayerID == -1) then
        return;
    end
    -- Update Header
    local playerTrade   :table  = Players[localPlayerID]:GetTrade();
    local routesActive  :number = playerTrade:GetNumOutgoingRoutes();
    local routesCapacity:number = playerTrade:GetOutgoingRouteCapacity();
Controls.HeaderLabel:SetText(Locale.ToUpper("LOC_TRADE_OVERVIEW_MY_ROUTES"));
    Controls.ActiveRoutesLabel:SetHide(false);
    -- If our active routes exceed our route capacity then color active route
number red
    local routesActiveText:string = ""
    if routesActive > routesCapacity then
        routesActiveText = "[COLOR_RED]" .. tostring(routesActive) ..
"[ENDCOLOR]";
    else
        routesActiveText = tostring(routesActive);
    end
    Controls.ActiveRoutesLabel:SetText(Locale.Lookup("LOC_TRADE_OVERVIEW_ACTIVE_
ROUTES", routesActiveText, routesCapacity));
    local localPlayerRunningRoutes:table = GetLocalPlayerRunningRoutes();
    -- Gather data and apply filter
    local routesSortedByPlayer:table = {};
    for _, route in ipairs(localPlayerRunningRoutes) do
        if m_filterList[m_filterSelected].FilterFunction and m_filterList[m_filt
```

```
erSelected].FilterFunction(Players[route.DestinationCityPlayer]) then
            -- Make sure we have a table for each destination player
            if routesSortedByPlayer[route.DestinationCityPlayer] == nil then
                routesSortedByPlayer[route.DestinationCityPlayer] = {};
            end
            table.insert(routesSortedByPlayer[route.DestinationCityPlayer],
route);
        end
    end
    -- Add routes to local player cities
    if routesSortedByPlayer[localPlayerID] ~= nil then
        CreatePlayerHeader(Players[localPlayerID]);
        routesSortedByPlayer[localPlayerID] =
SortTradeRoutes(routesSortedByPlayer[localPlayerID], m_GroupSortBySettings);
        for _, route in ipairs(routesSortedByPlayer[localPlayerID]) do
            AddRouteInstanceFromRouteInfo(route);
        end
    end
    -- Add routes to other civs
    local haveAddedCityStateHeader:boolean = false;
    for playerID, routes in pairs(routesSortedByPlayer) do
        if playerID ~= localPlayerID then
            routes = SortTradeRoutes(routes, m_GroupSortBySettings);
            -- Skip City States as these are added below
            local playerInfluence:table = Players[playerID]:GetInfluence();
            if not playerInfluence:CanReceiveInfluence() then
                CreatePlayerHeader(Players[playerID]);
                for _, route in ipairs(routes) do
                    AddRouteInstanceFromRouteInfo(route);
                end
            else
                -- Add city state routes
                if not haveAddedCityStateHeader then
                    haveAddedCityStateHeader = true;
                    CreateCityStateHeader();
                end
                for _, route in ipairs(routes) do
                    AddRouteInstanceFromRouteInfo(route);
                end
            end
        end
    end
    -- Determine how many unused routes we have
```

```
    local unusedRoutes  :number = routesCapacity - routesActive;
    if unusedRoutes > 0 then
        CreateUnusedRoutesHeader();
        local idleTradeUnits:table = GetIdleTradeUnits(localPlayerID);
        -- Assign idle trade units to unused routes
        for i=1, unusedRoutes, 1 do
            if #idleTradeUnits > 0 then
                -- Add button to choose a route for this trader
                AddChooseRouteButtonInstance(idleTradeUnits[1]);
                table.remove(idleTradeUnits, 1);
            else
                -- Add button to produce new trade unit
                AddProduceTradeUnitButtonInstance();
            end
        end
    end
end
-- Show Routes To My Cities Tab
function ViewRoutesToCities()
    -- Update Tabs
    SetMyRoutesTabSelected(false);
    SetRoutesToCitiesTabSelected(true);
    SetAvailableRoutesTabSelected(false);
    -- Update Header
    Controls.HeaderLabel:SetText(Locale.ToUpper("LOC_TRADE_OVERVIEW_ROUTES_TO_MY
_CITIES"));
    Controls.ActiveRoutesLabel:SetHide(true);
    -- Gather data
    local routesSortedByPlayer:table = {};
    local players = Game.GetPlayers{ Alive=true };
    for _, player in ipairs(players) do
        -- Don't show domestic routes
        if player:GetID() ~= Game.GetLocalPlayer() then
            if m_filterList[m_filterSelected].FilterFunction and
m_filterList[m_filterSelected].FilterFunction(player) then
                for _, city in player:GetCities():Members() do
                    local outgoingRoutes = city:GetTrade():GetOutgoingRoutes();
                    for _, route in ipairs(outgoingRoutes) do
                        -- Make sure the destination city is owned by the local
player
                        if route.DestinationCityPlayer == Game.GetLocalPlayer()
then
                            -- Make sure we have a table for each destination
```

```
player
                            if routesSortedByPlayer[route.OriginCityPlayer] ==
nil then
                                routesSortedByPlayer[route.OriginCityPlayer] =
{};
                            end
table.insert(routesSortedByPlayer[route.OriginCityPlayer], route);
                        end
                    end
                end
            end
        end
    end
    -- Add routes to stack
    for playerID, routes in pairs(routesSortedByPlayer) do
        CreatePlayerHeader(Players[playerID]);
        -- Sort the routes
        routes = SortTradeRoutes(routes, m_GroupSortBySettings);
        for _, route in ipairs(routes) do
            AddRouteInstanceFromRouteInfo(route);
        end
    end
end
-- Show Available Routes Tab
-- Note: There is a lot OPT prints and time information calculated
function ViewAvailableRoutes()
    -- Update Tabs
    SetMyRoutesTabSelected(false);
    SetRoutesToCitiesTabSelected(false);
    SetAvailableRoutesTabSelected(true);
    local localPlayerID = Game.GetLocalPlayer();
    if (localPlayerID == -1) then
        return;
    end
    local time1, time2;
    -- Update Header
    Controls.HeaderLabel:SetText(Locale.ToUpper("LOC_TRADE_OVERVIEW_AVAILABLE_RO
UTES"));
    Controls.ActiveRoutesLabel:SetHide(true);
    -- Dont rebuild if the turn has not advanced
    if (not m_HasBuiltTradeRouteTable) or Game.GetCurrentGameTurn() >
m_LastTurnBuiltTradeRouteTable then
        time1 = Automation.GetTime()
```

```
        RebuildAvailableTradeRoutesTable();
        time2 = Automation.GetTime()
        if dbug_print then
            print(string.format("Time taken to build routes: %.4f sec(s)",
time2-time1))
        end
        -- Cache routes info.
        time1 = Automation.GetTime()
        CacheEmpty();
        if CacheRoutesInfo(m_AvailableTradeRoutes) then
            time2 = Automation.GetTime()
            if dbug_print then
                print(string.format("Time taken to cache: %.4f sec(s)",
time2-time1))
            end
        end
        -- Just rebuilt base routes table. need to do everything again
        m_SortSettingsChanged = true;
        m_FilterSettingsChanged = true;
        m_GroupSettingsChanged = true;
    else
        if dbug_print then
            print("Trade Route table last built on: " ..
m_LastTurnBuiltTradeRouteTable .. ". Current game turn: " ..
Game.GetCurrentGameTurn());
        end
        if opt_print then
            print("OPT: Not Rebuilding or recaching routes table")
        end
    end
    -- Filter the routes here. This allows for max improvement in speed if a
filter is selected
    if m_FilterSettingsChanged then
        time1 = Automation.GetTime()
        m_FinalTradeRoutes = FilterTradeRoutes(m_AvailableTradeRoutes);
        time2 = Automation.GetTime()
        if dbug_print then
            print(string.format("Time taken to filter: %.4f sec(s)",
time2-time1))
        end
        -- Need to regroup routes (some groups could dissapear because of
filter)
        m_GroupSettingsChanged = true
```

```
    else
        if opt_print then
            print("OPT: Not refiltering routes")
        end
    end
    -- Sort and display the routes
    if not GroupSettingIsNone(m_groupBySelected) then
        -- Group routes. Use the filtered list of routes
        if m_GroupSettingsChanged then
            time1 = Automation.GetTime()
            m_AvailableGroupedRoutes = GroupRoutes(m_FinalTradeRoutes,
m_groupByList[m_groupBySelected].groupByID)
            time2 = Automation.GetTime()
            if dbug_print then
                print(string.format("Time taken to group: %.4f sec(s)",
time2-time1))
            end
            -- Need to resort to show correct order
            m_SortSettingsChanged = true
        else
            if opt_print then
                print("OPT: Not regrouping routes")
            end
        end
        -- Sort within each group, and then sort groups
        if m_SortSettingsChanged then
            -- Sort within each group
            time1 = Automation.GetTime()
            for i=1, #m_AvailableGroupedRoutes do
                m_AvailableGroupedRoutes[i] =
SortTradeRoutes(m_AvailableGroupedRoutes[i], m_InGroupSortBySettings)
            end
            time2 = Automation.GetTime()
            if dbug_print then
                print(string.format("Time taken to within group sort: %.4f
sec(s)", time2-time1))
            end
            -- Sort the order of groups. You need to do this AFTER each group
has been sorted
            time1 = Automation.GetTime()
            m_AvailableGroupedRoutes =
SortGroupedRoutes(m_AvailableGroupedRoutes, m_GroupSortBySettings);
            time2 = Automation.GetTime()
```

```
            if dbug_print then
                print(string.format("Time taken to group sort: %.4f sec(s)",
time2-time1))
            end
        else
            if opt_print then
                print("OPT: Not resorting within and of groups")
            end
        end
        -- Show the groups
        for i=1, #m_AvailableGroupedRoutes do
            if m_groupByList[m_groupBySelected].groupByID ==
GROUP_BY_SETTINGS.ORIGIN then
                local originPlayer:table =
Players[m_AvailableGroupedRoutes[i][1].OriginCityPlayer];
                local originCity:table =
originPlayer:GetCities():FindID(m_AvailableGroupedRoutes[i][1].OriginCityID);
                DisplayGroup(m_AvailableGroupedRoutes[i], originCity);
            elseif m_groupByList[m_groupBySelected].groupByID ==
GROUP_BY_SETTINGS.DESTINATION then
                local destinationPlayer:table =
Players[m_AvailableGroupedRoutes[i][1].DestinationCityPlayer];
                local destinationCity:table = destinationPlayer:GetCities():Find
ID(m_AvailableGroupedRoutes[i][1].DestinationCityID);
                DisplayGroup(m_AvailableGroupedRoutes[i], destinationCity);
            end
        end
    else
        if m_FinalTradeRoutes ~= nil then
            if m_SortSettingsChanged or m_GroupSettingsChanged then
                time1 = Automation.GetTime()
                m_FinalTradeRoutes = SortTradeRoutes(m_FinalTradeRoutes,
m_GroupSortBySettings);
                time2 = Automation.GetTime()
                if dbug_print then
                    print(string.format("Time taken to sort: %.4f sec(s)",
time2-time1))
                end
            else
                if opt_print then
                    print("OPT: Not resorting routes")
                end
            end
```

```
            AddRouteInstancesFromTable(m_FinalTradeRoutes);
        end
    end
    -- Everything is done if it reaches here
    m_SortSettingsChanged = false;
    m_FilterSettingsChanged = false;
    m_GroupSettingsChanged = false;
end
function DisplayGroup(routesTable:table, city:table)
    -- dump(routesTable[1])
    local routeCount:number = #routesTable;
    if routeCount > 0 then
        -- Find if the city is in exclusion list
        local cityEntry:table = {
            OwnerID = city:GetOwner(),
            CityID = city:GetID()
        };
        local groupExpandIndex = findIndex(m_GroupsFullyExpanded, cityEntry,
CompareCityEntries);
        local groupCollapseIndex = findIndex(m_GroupsFullyCollapsed, cityEntry,
CompareCityEntries);
        if dbug_print then
            -- print(Locale.Lookup(city:GetName()) .. ": " .. groupExpandIndex
.. " " .. groupCollapseIndex )
        end
        if (groupExpandIndex > 0) then
            CreateCityHeader(city, routeCount, routeCount, "");
            AddRouteInstancesFromTable(routesTable);
        elseif (groupCollapseIndex > 0) then
            CreateCityHeader(city, 0, routeCount,
GetCityHeaderTooltipString(routesTable[1]));
            AddRouteInstancesFromTable(routesTable, 0);
        else
            if m_GroupExpandAll then
                -- If showing all, add city to expand list, and display all
                table.insert(m_GroupsFullyExpanded, cityEntry);
                CreateCityHeader(city, routeCount, routeCount, "");
                AddRouteInstancesFromTable(routesTable);
            elseif m_GroupCollapseAll then
                -- If hiding all, add city to collapse list, and hide it
                table.insert(m_GroupsFullyCollapsed, cityEntry);
                CreateCityHeader(city, 0, routeCount,
GetCityHeaderTooltipString(routesTable[1]));
```

```
                AddRouteInstancesFromTable(routesTable, 0);
            else
                CreateCityHeader(city,
math.min(SEMI_EXPAND_SETTINGS[m_groupBySelected], routeCount), routeCount, "");
                AddRouteInstancesFromTable(routesTable,
SEMI_EXPAND_SETTINGS[m_groupBySelected]);
            end
        end
    end
end
-- ---------------------------------------------------------------------------
-- Tab UI Helpers
-- ---------------------------------------------------------------------------
function SetMyRoutesTabSelected( isSelected:boolean )
    Controls.MyRoutesButton:SetSelected(isSelected);
    Controls.MyRoutesTabLabel:SetHide(isSelected);
    Controls.MyRoutesSelected:SetHide(not isSelected);
    Controls.MyRoutesSelectedArrow:SetHide(not isSelected);
    Controls.MyRoutesTabSelectedLabel:SetHide(not isSelected);
end
function SetRoutesToCitiesTabSelected( isSelected:boolean )
    Controls.RoutesToCitiesButton:SetSelected(isSelected);
    Controls.RoutesToCitiesTabLabel:SetHide(isSelected);
    Controls.RoutesToCitiesSelected:SetHide(not isSelected);
    Controls.RoutesToCitiesSelectedArrow:SetHide(not isSelected);
    Controls.RoutesToCitiesTabSelectedLabel:SetHide(not isSelected);
end
function SetAvailableRoutesTabSelected( isSelected:boolean )
    Controls.AvailableRoutesButton:SetSelected(isSelected);
    Controls.AvailableRoutesTabLabel:SetHide(isSelected);
    Controls.AvailableRoutesSelected:SetHide(not isSelected);
    Controls.AvailableRoutesSelectedArrow:SetHide(not isSelected);
    Controls.AvailableRoutesTabSelectedLabel:SetHide(not isSelected);
end
function GetCityHeaderTooltipString( routeInfo:table )
    return "Top Route: " .. GetTradeRouteString(routeInfo) .. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_TOOLTIP_BREAKER")
                .. "[NEWLINE]" .. GetTradeRouteYieldString(routeInfo);
end
-- ===========================================================================
--  Route Instance Creators
-- ===========================================================================
function AddChooseRouteButtonInstance( tradeUnit:table )
```

```
    local simpleButtonInstance:table = m_SimpleButtonInstanceIM:GetInstance();
    simpleButtonInstance.GridButton:SetText(Locale.Lookup("LOC_TRADE_OVERVIEW_CH
OOSE_ROUTE"));
    simpleButtonInstance.GridButton:SetDisabled(false);
    simpleButtonInstance.GridButton:RegisterCallback( Mouse.eLClick,
        function()
            SelectUnit( tradeUnit );
        end
    );
end
function AddProduceTradeUnitButtonInstance()
    local simpleButtonInstance:table = m_SimpleButtonInstanceIM:GetInstance();
    simpleButtonInstance.GridButton:SetText(Locale.Lookup("LOC_TRADE_OVERVIEW_PR
ODUCE_TRADE_UNIT"));
    simpleButtonInstance.GridButton:SetDisabled(true);
end
function AddRouteInstancesFromTable( tradeRoutes:table, showCount:number )
    if showCount then
        local len = math.min(showCount, #tradeRoutes)
        for i=1, len do
            AddRouteInstanceFromRouteInfo(tradeRoutes[i]);
        end
    else
        local tTime = Automation.GetTime();
        for i=1, #tradeRoutes do
            if (tTime + 1 < Automation.GetTime()) then
                if dbug_print then
                    print("+1 sec ... " .. i)
                end
                tTime = Automation.GetTime()
            end
            AddRouteInstanceFromRouteInfo(tradeRoutes[i]);
        end
    end
end
function AddRouteInstanceFromRouteInfo( routeInfo:table )
    -- Get all the info, to build the route
    local originPlayer:table = Players[routeInfo.OriginCityPlayer];
    local originCity:table =
originPlayer:GetCities():FindID(routeInfo.OriginCityID);
    local destinationPlayer:table = Players[routeInfo.DestinationCityPlayer];
    local destinationCity:table =
destinationPlayer:GetCities():FindID(routeInfo.DestinationCityID);
```

```
    local routeInstance:table = m_RouteInstanceIM:GetInstance();
    local destinationBackColor, destinationFrontColor, darkerBackColor,
brighterBackColor = GetPlayerColorInfo(routeInfo.DestinationCityPlayer, true);
    local originBackColor, originFrontColor =
GetPlayerColorInfo(routeInfo.OriginCityPlayer, true);
    local tintBackColor = DarkenLightenColor(destinationBackColor,
tintColorOffset, tintColorOpacity);
    -- Update colors
    if tintRouteEntry then
        routeInstance.GridButton:SetColor(tintBackColor);
    end
    routeInstance.TurnsToComplete:SetColor(destinationFrontColor);
    routeInstance.BannerBase:SetColor(destinationBackColor);
    routeInstance.BannerDarker:SetColor(darkerBackColor);
    routeInstance.BannerLighter:SetColor(brighterBackColor);
    routeInstance.RouteLabel:SetColor(destinationFrontColor);
    -- Update Route Label
    routeInstance.RouteLabel:SetText(Locale.ToUpper(originCity:GetName()) .. " "
.. Locale.ToUpper("LOC_TRADE_OVERVIEW_TO") .. " " ..
Locale.ToUpper(destinationCity:GetName()));
    -- Update yield directional arrows
    routeInstance.OriginCivArrow:SetColor(originFrontColor);
    routeInstance.DestinationCivArrow:SetColor(destinationFrontColor);
    SetOriginRouteInstanceYields(routeInstance, routeInfo)
    if GetNetYieldForDestinationCity(routeInfo, true) > 0 then
        if dbug_print then
            print(GetTradeRouteString(routeInfo), "has destination has yield")
        end
        routeInstance.DestinationYields:SetHide(false);
        SetDestinationRouteInstanceYields(routeInstance, routeInfo)
    else
        routeInstance.DestinationYields:SetHide(true);
    end
    -- Update City State Quest Icon
    routeInstance.CityStateQuestIcon:SetHide(true);
    local questTooltip  : string = Locale.Lookup("LOC_CITY_STATES_QUESTS");
    local tradeRouteQuestInfo:table = GameInfo.Quests["QUEST_SEND_TRADE_ROUTE"];
    local questsManager:table = Game.GetQuestsManager();
    if IsCityStateWithTradeQuest(destinationPlayer) then
        questTooltip = questTooltip .. "[NEWLINE]" ..
tradeRouteQuestInfo.IconString ..
questsManager:GetActiveQuestName(routeInfo.OriginCityPlayer,
routeInfo.DestinationCityPlayer, tradeRouteQuestInfo.Index);
```

```
        routeInstance.CityStateQuestIcon:SetHide(false);
        routeInstance.CityStateQuestIcon:SetToolTipString(questTooltip);
    end
    -- Update Diplomatic Visibility
    routeInstance.VisibilityBonusGrid:SetHide(false);
    routeInstance.TourismBonusGrid:SetHide(false);
    -- TODO - Can we make this simpler?
    -- Do we display the tourism or visibilty bonus? Hide them if we are showing
them somewhere else, or it is a city state, or it is domestic route
    if IsCityState(destinationPlayer) or routeInfo.OriginCityPlayer ==
routeInfo.DestinationCityPlayer
        or m_groupByList[m_groupBySelected].groupByID ==
GROUP_BY_SETTINGS.DESTINATION or m_currentTab ~= TRADE_TABS.AVAILABLE_ROUTES
then
        routeInstance.VisibilityBonusGrid:SetHide(true);
        routeInstance.TourismBonusGrid:SetHide(true);
        -- Also hide the trading post if grouping by destination (will be shown
in the header)
        if m_groupByList[m_groupBySelected].groupByID ==
GROUP_BY_SETTINGS.DESTINATION then
            routeInstance.TradingPostIndicator:SetHide(true);
        elseif not hideTradingPostIcon then
            routeInstance.TradingPostIndicator:SetHide(false);
        end
    else
        -- Determine are diplomatic visibility status
        local visibilityIndex:number =
GetVisibilityIndex(routeInfo.DestinationCityPlayer, true)
        -- Determine this player has a trade route with the local player
        local hasTradeRoute:boolean =
GetHasActiveRoute(routeInfo.DestinationCityPlayer, true)
        -- Display trade route tourism modifier
        local extraTourismModifier =
originPlayer:GetCulture():GetExtraTradeRouteTourismModifier();
        -- TODO: Use LOC_TRADE_OVERVIEW_TOURISM_BONUS when we can update the
text
        routeInstance.TourismBonusPercentage:SetText("+" ..
Locale.ToPercent((BASE_TOURISM_MODIFIER + extraTourismModifier)/100));
        if hasTradeRoute then
routeInstance.TourismBonusPercentage:SetColorByName("TradeOverviewTextCS");
routeInstance.TourismBonusIcon:SetTexture(0,0,"Tourism_VisitingSmall");
            routeInstance.TourismBonusGrid:LocalizeAndSetToolTip("LOC_TRADE_OVER
VIEW_TOOLTIP_TOURISM_BONUS");
```

```
routeInstance.VisibilityBonusIcon:SetTexture("Diplomacy_VisibilityIcons");
            routeInstance.VisibilityBonusIcon:SetVisState(Clamp(visibilityIndex
- 1, 0, 3));
            routeInstance.VisibilityBonusGrid:LocalizeAndSetToolTip("LOC_TRADE_O
VERVIEW_TOOLTIP_DIPLOMATIC_VIS_BONUS");
        else
            routeInstance.TourismBonusPercentage:SetColorByName("TradeOverviewTe
xtDisabledCS");
routeInstance.TourismBonusIcon:SetTexture(0,0,"Tourism_VisitingSmallGrey");
            routeInstance.TourismBonusGrid:LocalizeAndSetToolTip("LOC_TRADE_OVER
VIEW_TOOLTIP_NO_TOURISM_BONUS");
routeInstance.VisibilityBonusIcon:SetTexture("Diplomacy_VisibilityIconsGrey");
            routeInstance.VisibilityBonusIcon:SetVisState(Clamp(visibilityIndex,
0, 3));
            routeInstance.VisibilityBonusGrid:LocalizeAndSetToolTip("LOC_TRADE_O
VERVIEW_TOOLTIP_NO_DIPLOMATIC_VIS_BONUS");
        end
    end
    -- Update Trading Post Icon
    if GroupSettingIsNone(m_groupBySelected) or m_groupBySelected ==
GROUP_BY_SETTINGS.ORIGIN then
        routeInstance.TradingPostIndicator:SetHide(false);
    else
        routeInstance.TradingPostIndicator:SetHide(true);
    end
    if GetRouteHasTradingPost(routeInfo, true) then
        routeInstance.TradingPostIndicator:SetAlpha(1.0);
        routeInstance.TradingPostIndicator:LocalizeAndSetToolTip("LOC_TRADE_OVER
VIEW_TOOLTIP_TRADE_POST_ESTABLISHED");
    else
        routeInstance.TradingPostIndicator:SetAlpha(0.2);
        routeInstance.TradingPostIndicator:LocalizeAndSetToolTip("LOC_TRADE_OVER
VIEW_TOOLTIP_NO_TRADE_POST");
    end
    -- Update turns to complete route
    local tooltipString:string;
    local tradePathLength, tripsToDestination, turnsToCompleteRoute =
GetRouteInfo(routeInfo, true);
    if routeInfo.TurnsRemaining ~= nil then
        routeInstance.TurnsToComplete:SetText(routeInfo.TurnsRemaining);
        tooltipString = (
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_ALT_HELP_TOOLTIP",
routeInfo.TurnsRemaining) .. "[NEWLINE]" ..
```

```
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_TOOLTIP_BREAKER") .. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_ROUTE_LENGTH_TOOLTIP", tradePathLength)
.. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_TRIPS_COUNT_TOOLTIP",
tripsToDestination) .. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_TOOLTIP",
(Game.GetCurrentGameTurn() + routeInfo.TurnsRemaining)) );
    elseif m_currentTab == TRADE_TABS.ROUTES_TO_CITIES then
        routeInstance.TurnsToComplete:SetText(turnsToCompleteRoute);
        tooltipString = (
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_HELP_TOOLTIP") .. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_TOOLTIP_BREAKER") .. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_ROUTE_LENGTH_TOOLTIP", tradePathLength)
.. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_TRIPS_COUNT_TOOLTIP",
tripsToDestination) );
    else
        routeInstance.TurnsToComplete:SetText(turnsToCompleteRoute);
        tooltipString = (
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_HELP_TOOLTIP") .. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_TOOLTIP_BREAKER") .. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_ROUTE_LENGTH_TOOLTIP", tradePathLength)
.. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_TRIPS_COUNT_TOOLTIP",
tripsToDestination) .. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_ALT_TOOLTIP",
turnsToCompleteRoute, (Game.GetCurrentGameTurn() + turnsToCompleteRoute)) );
    end
    routeInstance.TurnsToComplete:SetToolTipString( tooltipString );
    local originTextureOffsetX, originTextureOffsetY, originTextureSheet,
originTooltip = GetPlayerIconInfo(routeInfo.OriginCityPlayer, true)
    local destinationTextureOffsetX, destinationTextureOffsetY,
destinationTextureSheet, destinationTooltip =
GetPlayerIconInfo(routeInfo.DestinationCityPlayer, true)
    -- Origin Civ Icon
    routeInstance.OriginCivIcon:SetTexture(originTextureOffsetX,
originTextureOffsetY, originTextureSheet);
    routeInstance.OriginCivIcon:LocalizeAndSetToolTip(originTooltip);
    routeInstance.OriginCivIcon:SetColor(originFrontColor);
    routeInstance.OriginCivIconBacking:SetColor(originBackColor);
    -- Destination Civ Icon
    routeInstance.DestinationCivIcon:SetTexture(destinationTextureOffsetX,
destinationTextureOffsetY, destinationTextureSheet);
```

```
    routeInstance.DestinationCivIcon:SetColor(destinationFrontColor);
    routeInstance.DestinationCivIconBacking:SetColor(destinationBackColor);
    routeInstance.DestinationCivIcon:LocalizeAndSetToolTip(destinationTooltip);
    -- Hide the cancel automation button by default
    routeInstance.CancelAutomation:SetHide(true);
    -- Should we display the cancel automation?
    if m_currentTab == TRADE_TABS.MY_ROUTES and routeInfo.TraderUnitID ~= nil
then
        if IsTraderAutomated(routeInfo.TraderUnitID) then
            -- Unhide the cancel automation
            routeInstance.CancelAutomation:SetHide(false);
            -- Add button callback
            routeInstance.CancelAutomation:RegisterCallback( Mouse.eLClick,
                function()
                    CancelAutomatedTrader(routeInfo.TraderUnitID);
                    Refresh();
                end
            );
        end
    end
    if routeInfo.TraderUnitID then
        local tradeUnit:table =
originPlayer:GetUnits():FindID(routeInfo.TraderUnitID);
        routeInstance.GridButton:RegisterCallback( Mouse.eLClick,
            function()
                SelectUnit(tradeUnit);
            end
        );
    -- Add button hookups for only this tab
    elseif m_currentTab == TRADE_TABS.AVAILABLE_ROUTES and m_AvailableTraders ~=
nil and table.count(m_AvailableTraders) > 0 then
        -- Check if we have free trader in that city
        if m_AvailableTraders[routeInfo.OriginCityID] ~= nil and
table.count(m_AvailableTraders[routeInfo.OriginCityID]) > 0 then
            -- Get first trader
            local traderID = m_AvailableTraders[routeInfo.OriginCityID][1]
            local tradeUnit:table = originPlayer:GetUnits():FindID(traderID);
            routeInstance.GridButton:RegisterCallback( Mouse.eLClick,
                function()
                    SelectFreeTrader(tradeUnit, routeInfo.DestinationCityPlayer,
routeInfo.DestinationCityID);
                end
            );
```

```
        else -- Cycle through all free traders and open transfer-to screen for
them
            local co = coroutine.create(
                function()
                    while true do -- Infinitely cycle
                        -- Do we have traders to cycle between?
                        if CountTraders(m_AvailableTraders) > 0 then
                            for cityID in pairs(m_AvailableTraders) do
                                for i in pairs(m_AvailableTraders[cityID]) do
                                    local traderID =
m_AvailableTraders[cityID][i]
                                    local tradeUnit:table =
originPlayer:GetUnits():FindID(traderID);
                                    if dbug_print then
                                        print("Calling transfer from " ..
cityID)
                                    end
                                    TransferTraderTo(tradeUnit, originCity)
                                    coroutine.yield()
                                end
                            end
                        else
                            if dbug_print then
                                print("Backup 2 yield")
                            end
                            coroutine.yield() -- gauranteed yield to prevent
infinite cycle bug
                        end
                    end
                end
            );
            routeInstance.GridButton:RegisterCallback( Mouse.eLClick,
                function()
                    CycleTraders(co)
                end
            );
        end
    end
end
-- ---------------------------------------------------------------------------
-- Route button helpers
-- ---------------------------------------------------------------------------
function SetOriginRouteInstanceYields(routeInstance, routeInfo)
```

```
    local yieldTexts = {}
    for yieldIndex = START_INDEX, END_INDEX do
        local yieldAmount = GetYieldForOriginCity(yieldIndex, routeInfo, true)
        yieldAmount = Round(yieldAmount, 1)
        local iconString, text = FormatYieldText(yieldIndex, yieldAmount)
        yieldTexts[yieldIndex] = text .. iconString
    end
    routeInstance.OriginYieldFoodLabel:SetText(yieldTexts[FOOD_INDEX])
routeInstance.OriginYieldProductionLabel:SetText(yieldTexts[PRODUCTION_INDEX])
    routeInstance.OriginYieldGoldLabel:SetText(yieldTexts[GOLD_INDEX])
    routeInstance.OriginYieldScienceLabel:SetText(yieldTexts[SCIENCE_INDEX])
    routeInstance.OriginYieldCultureLabel:SetText(yieldTexts[CULTURE_INDEX])
    routeInstance.OriginYieldFaithLabel:SetText(yieldTexts[FAITH_INDEX])
end
function SetDestinationRouteInstanceYields(routeInstance, routeInfo)
    local yieldTexts = {}
    for yieldIndex = START_INDEX, END_INDEX do
        local yieldAmount = GetYieldForDestinationCity(yieldIndex, routeInfo,
true)
        yieldAmount = Round(yieldAmount, 1)
        local iconString, text = FormatYieldText(yieldIndex, yieldAmount)
        yieldTexts[yieldIndex] = text .. iconString
    end
    routeInstance.DestinationYieldFoodLabel:SetText(yieldTexts[FOOD_INDEX])
    routeInstance.DestinationYieldProductionLabel:SetText(yieldTexts[PRODUCTION_
INDEX])
    routeInstance.DestinationYieldGoldLabel:SetText(yieldTexts[GOLD_INDEX])
routeInstance.DestinationYieldScienceLabel:SetText(yieldTexts[SCIENCE_INDEX])
routeInstance.DestinationYieldCultureLabel:SetText(yieldTexts[CULTURE_INDEX])
    routeInstance.DestinationYieldFaithLabel:SetText(yieldTexts[FAITH_INDEX])
end
-- ===========================================================================
--  Header Instance Creators
-- ===========================================================================
function CreateSectionDivider()
    if m_dividerCount > 0 then
        local dividerInstance:table = m_DividerInstanceIM:GetInstance();
    end
    m_dividerCount = m_dividerCount + 1
end
function CreatePlayerHeader( player:table )
    CreateSectionDivider()
    local headerInstance:table = m_HeaderInstanceIM:GetInstance();
```

```
    local playerID = player:GetID()
    local pPlayerConfig:table = PlayerConfigurations[playerID];
    headerInstance.HeaderLabel:SetText(Locale.ToUpper(pPlayerConfig:GetPlayerNam
e()));
    -- If the current tab is not available routes, hide the collapse button, and
trading post
    if m_currentTab ~= TRADE_TABS.AVAILABLE_ROUTES then
        headerInstance.RoutesExpand:SetHide(true);
        headerInstance.RouteCountLabel:SetHide(true);
        headerInstance.TradingPostIndicator:SetHide(true);
    end
    if colorCityPlayerHeader then
        headerInstance.CityBannerFill:SetHide(false);
        local backColor, frontColor = GetPlayerColorInfo(playerID, true);
        headerBackColor = DarkenLightenColor(backColor, backdropColorOffset,
backdropColorOpacity);
        headerFrontColor = DarkenLightenColor(frontColor, labelColorOffset,
labelColorOpacity);
        gridBackColor = DarkenLightenColor(backColor, backdropGridColorOffset,
backdropGridColorOpacity);
        headerInstance.CityBannerFill:SetColor( headerBackColor );
        headerInstance.HeaderLabel:SetColor(headerFrontColor);
        headerInstance.HeaderGrid:SetColor(gridBackColor);
    else
        -- Hide the colored UI elements
        headerInstance.CityBannerFill:SetHide(true);
    end
    -- If not local player or a city state
    if (playerID ~=  Game.GetLocalPlayer() and (not IsCityState(player))) then
        -- Determine are diplomatic visibility status
        headerInstance.TourismBonusGrid:SetHide(false);
        headerInstance.VisibilityBonusGrid:SetHide(false)
        local visibilityIndex:number = GetVisibilityIndex(playerID, true)
        -- Determine this player has a trade route with the local player
        local hasTradeRoute:boolean = GetHasActiveRoute(playerID, true)
        -- Display trade route tourism modifier
        local extraTourismModifier =
Players[Game.GetLocalPlayer()]:GetCulture():GetExtraTradeRouteTourismModifier();
        -- TODO: Use LOC_TRADE_OVERVIEW_TOURISM_BONUS when we can update the
text
        headerInstance.TourismBonusPercentage:SetText("+" ..
Locale.ToPercent((BASE_TOURISM_MODIFIER + extraTourismModifier)/100));
        if hasTradeRoute then
```

```
headerInstance.TourismBonusPercentage:SetColorByName("TradeOverviewTextCS");
headerInstance.TourismBonusIcon:SetTexture(0,0,"Tourism_VisitingSmall");
            headerInstance.TourismBonusGrid:LocalizeAndSetToolTip("LOC_TRADE_OVE
RVIEW_TOOLTIP_TOURISM_BONUS");
headerInstance.VisibilityBonusIcon:SetTexture("Diplomacy_VisibilityIcons");
headerInstance.VisibilityBonusIcon:SetVisState(math.min(math.max(visibilityIndex
- 1, 0), 3));
            headerInstance.VisibilityBonusGrid:LocalizeAndSetToolTip("LOC_TRADE_
OVERVIEW_TOOLTIP_DIPLOMATIC_VIS_BONUS");
        else
            headerInstance.TourismBonusPercentage:SetColorByName("TradeOverviewT
extDisabledCS");
headerInstance.TourismBonusIcon:SetTexture(0,0,"Tourism_VisitingSmallGrey");
            headerInstance.TourismBonusGrid:LocalizeAndSetToolTip("LOC_TRADE_OVE
RVIEW_TOOLTIP_NO_TOURISM_BONUS");
headerInstance.VisibilityBonusIcon:SetTexture("Diplomacy_VisibilityIconsGrey");
            headerInstance.VisibilityBonusIcon:SetVisState(math.min(math.max(vis
ibilityIndex, 0), 3));
            headerInstance.VisibilityBonusGrid:LocalizeAndSetToolTip("LOC_TRADE_
OVERVIEW_TOOLTIP_NO_DIPLOMATIC_VIS_BONUS");
        end
    else
        if dbug_print then
            -- print("Not displaying vis bonuses")
        end
        headerInstance.TourismBonusGrid:SetHide(true);
        headerInstance.VisibilityBonusGrid:SetHide(true);
    end
end
function CreateCityStateHeader()
    CreateSectionDivider()
    local headerInstance:table = m_HeaderInstanceIM:GetInstance();
    -- If the current tab is not available routes, hide the collapse button, and
trading post
    if m_currentTab ~= TRADE_TABS.AVAILABLE_ROUTES then
        headerInstance.RoutesExpand:SetHide(true);
        headerInstance.RouteCountLabel:SetHide(true);
        headerInstance.TradingPostIndicator:SetHide(true);
    end
    -- Reset Color for city states
    headerInstance.HeaderGrid:SetColor(0xFF666666);
    headerInstance.CityBannerFill:SetHide(true);
    headerInstance.HeaderLabel:SetColorByName("Beige");
```

```
    headerInstance.HeaderLabel:SetText(Locale.ToUpper("LOC_TRADE_OVERVIEW_CITY_S
TATES"));
    headerInstance.VisibilityBonusGrid:SetHide(true);
    headerInstance.TourismBonusGrid:SetHide(true);
end
function CreateUnusedRoutesHeader()
    CreateSectionDivider()
    local headerInstance:table = m_HeaderInstanceIM:GetInstance();
    headerInstance.HeaderLabel:SetText(Locale.ToUpper("LOC_TRADE_OVERVIEW_UNUSED
_ROUTES"));
    -- Reset Color for city states
    headerInstance.HeaderGrid:SetColor(0xFF666666);
    headerInstance.CityBannerFill:SetHide(true);
    headerInstance.HeaderLabel:SetColorByName("Beige");
    headerInstance.RoutesExpand:SetHide(true);
    headerInstance.RouteCountLabel:SetHide(true);
    headerInstance.TradingPostIndicator:SetHide(true);
    headerInstance.VisibilityBonusGrid:SetHide(true);
    headerInstance.TourismBonusGrid:SetHide(true);
end
function CreateCityHeader( city:table , currentRouteShowCount:number,
totalRoutes:number, tooltipString:string )
    CreateSectionDivider()
    local headerInstance:table = m_HeaderInstanceIM:GetInstance();
    local playerID:number = city:GetOwner();
    local pPlayer = Players[playerID];
    headerInstance.HeaderLabel:SetText(Locale.ToUpper(city:GetName()));
    if tooltipString ~= nil then
        headerInstance.HeaderGrid:SetToolTipString(tooltipString);
    end
    if m_currentTab == TRADE_TABS.AVAILABLE_ROUTES then
        headerInstance.RoutesExpand:SetHide(false);
        headerInstance.RouteCountLabel:SetHide(false);
        headerInstance.TradingPostIndicator:SetHide(false);
    end
    headerInstance.RouteCountLabel:SetText(currentRouteShowCount .. " / " ..
totalRoutes);
    -- If grouping by destination, show and refresh bonuses
    if m_groupByList[m_groupBySelected].groupByID ==
GROUP_BY_SETTINGS.DESTINATION then
        -- Update Trading Post Icon
        headerInstance.TradingPostIndicator:SetHide(false);
        if city:GetTrade():HasActiveTradingPost(Players[Game.GetLocalPlayer()])
```

```
then
            headerInstance.TradingPostIndicator:SetAlpha(1.0);
            headerInstance.TradingPostIndicator:LocalizeAndSetToolTip("LOC_TRADE
_OVERVIEW_TOOLTIP_TRADE_POST_ESTABLISHED");
        else
            headerInstance.TradingPostIndicator:SetAlpha(0.2);
            headerInstance.TradingPostIndicator:LocalizeAndSetToolTip("LOC_TRADE
_OVERVIEW_TOOLTIP_NO_TRADE_POST");
        end
        -- Update Diplomatic Visibility
        headerInstance.VisibilityBonusGrid:SetHide(false);
        headerInstance.TourismBonusGrid:SetHide(false);
        -- Do we display the tourism or visibilty bonus? Hide them if it is a
city state, or it is domestic route
        if IsCityState(pPlayer) or pPlayer:GetID() == Game.GetLocalPlayer() then
            headerInstance.VisibilityBonusGrid:SetHide(true);
            headerInstance.TourismBonusGrid:SetHide(true);
        else
            -- Determine are diplomatic visibility status
            local visibilityIndex:number = GetVisibilityIndex(playerID, true)
            -- Determine this player has a trade route with the local player
            local hasTradeRoute:boolean = GetHasActiveRoute(playerID, true)
            -- Display trade route tourism modifier
            local extraTourismModifier =
Players[Game.GetLocalPlayer()]:GetCulture():GetExtraTradeRouteTourismModifier();
            -- TODO: Use LOC_TRADE_OVERVIEW_TOURISM_BONUS when we can update the
text
            headerInstance.TourismBonusPercentage:SetText("+" ..
Locale.ToPercent((BASE_TOURISM_MODIFIER + extraTourismModifier)/100));
            if hasTradeRoute then
headerInstance.TourismBonusPercentage:SetColorByName("TradeOverviewTextCS");
headerInstance.TourismBonusIcon:SetTexture(0,0,"Tourism_VisitingSmall");
                headerInstance.TourismBonusGrid:LocalizeAndSetToolTip("LOC_TRADE
_OVERVIEW_TOOLTIP_TOURISM_BONUS");
headerInstance.VisibilityBonusIcon:SetTexture("Diplomacy_VisibilityIcons");
headerInstance.VisibilityBonusIcon:SetVisState(math.min(math.max(visibilityIndex
- 1, 0), 3));
                headerInstance.VisibilityBonusGrid:LocalizeAndSetToolTip("LOC_TR
ADE_OVERVIEW_TOOLTIP_DIPLOMATIC_VIS_BONUS");
            else
                headerInstance.TourismBonusPercentage:SetColorByName("TradeOverv
iewTextDisabledCS");
headerInstance.TourismBonusIcon:SetTexture(0,0,"Tourism_VisitingSmallGrey");
```

```
                headerInstance.TourismBonusGrid:LocalizeAndSetToolTip("LOC_TRADE
_OVERVIEW_TOOLTIP_NO_TOURISM_BONUS");
headerInstance.VisibilityBonusIcon:SetTexture("Diplomacy_VisibilityIconsGrey");
                headerInstance.VisibilityBonusIcon:SetVisState(math.min(math.max
(visibilityIndex, 0), 3));
                headerInstance.VisibilityBonusGrid:LocalizeAndSetToolTip("LOC_TR
ADE_OVERVIEW_TOOLTIP_NO_DIPLOMATIC_VIS_BONUS");
            end
        end
    else
        headerInstance.TourismBonusGrid:SetHide(true);
        headerInstance.VisibilityBonusGrid:SetHide(true);
        headerInstance.TradingPostIndicator:SetHide(true);
    end
    local cityEntry:table = {
        OwnerID = playerID,
        CityID = city:GetID()
    };
    local cityExclusionIndex = findIndex(m_GroupsFullyExpanded, cityEntry,
CompareCityEntries);
    if cityExclusionIndex == -1 then
        headerInstance.RoutesExpand:SetCheck(false);
        headerInstance.RoutesExpand:SetCheckTextureOffsetVal(0,0);
    else
        headerInstance.RoutesExpand:SetCheck(true);
        headerInstance.RoutesExpand:SetCheckTextureOffsetVal(0,22);
    end
    headerInstance.RoutesExpand:RegisterCallback( Mouse.eLClick, function()
OnExpandRoutes(headerInstance.RoutesExpand, city:GetOwner(), city:GetID()); end
);
    headerInstance.RoutesExpand:RegisterCallback( Mouse.eMouseEnter, function()
UI.PlaySound("Main_Menu_Mouse_Over"); end);
    headerInstance.RoutesExpand:RegisterCallback( Mouse.eRClick, function()
OnCollapseRoutes(headerInstance.RoutesExpand, city:GetOwner(), city:GetID());
end );
    headerInstance.RoutesExpand:RegisterCallback( Mouse.eMouseEnter, function()
UI.PlaySound("Main_Menu_Mouse_Over"); end);
    if colorCityPlayerHeader then
        headerInstance.CityBannerFill:SetHide(false);
        local backColor, frontColor = GetPlayerColorInfo(playerID, true);
        headerBackColor = DarkenLightenColor(backColor, backdropColorOffset,
backdropColorOpacity);
        headerFrontColor = DarkenLightenColor(frontColor, labelColorOffset,
```

```
labelColorOpacity);
        gridBackColor = DarkenLightenColor(backColor, backdropGridColorOffset,
backdropGridColorOpacity);
        headerInstance.HeaderLabel:SetColor(headerFrontColor);
        headerInstance.CityBannerFill:SetColor(headerBackColor);
        headerInstance.HeaderGrid:SetColor(gridBackColor);
    else
        -- Hide the colored UI elements
        headerInstance.CityBannerFill:SetHide(true);
    end
end
function OnExpandRoutes( checkbox, cityOwnerID:number, cityID:number )
    if m_GroupCollapseAll then
        m_GroupCollapseAll = false;
        Controls.GroupCollapseAllCheckBox:SetCheck(false);
    end
    -- For some reason the Uncheck texture does not apply, so I had to hard code
the offset in.
    -- TODO: Find a fix for this
    if (checkbox:IsChecked()) then
        checkbox:SetCheckTextureOffsetVal(0,22);
        local cityEntry = {
            OwnerID = cityOwnerID,
            CityID = cityID
        };
        -- Only add entry if it isn't already in the list
        if findIndex(m_GroupsFullyExpanded, cityEntry, CompareCityEntries) == -1
then
            if dbug_print then
                print("Adding " .. GetCityEntryString(cityEntry) .. " to the
exclusion list");
            end
            table.insert(m_GroupsFullyExpanded, cityEntry);
        else
            if dbug_print then
                print("City already exists in exclusion list");
            end
        end
    else
        if m_GroupExpandAll then
            m_GroupExpandAll = false;
            Controls.GroupExpandAllCheckBox:SetCheck(false);
        end
```

```
        checkbox:SetCheckTextureOffsetVal(0,0);
        local cityEntry = {
            OwnerID = cityOwnerID,
            CityID = cityID
        };
        local cityIndex = findIndex(m_GroupsFullyExpanded, cityEntry,
CompareCityEntries)
        if findIndex(m_GroupsFullyExpanded, cityEntry, CompareCityEntries) > 0
then
            if dbug_print then
                print("Removing " .. GetCityEntryString(cityEntry) .. " to the
exclusion list");
            end
            table.remove(m_GroupsFullyExpanded, cityIndex);
        else
            if dbug_print then
                print("City does not exist in exclusion list");
            end
        end
    end
    Refresh();
end
function OnCollapseRoutes( checkbox, cityOwnerID:number, cityID:number )
    if m_GroupExpandAll then
        m_GroupExpandAll = false;
        Controls.GroupExpandAllCheckBox:SetCheck(false);
    end
    checkbox:SetCheck(false);
    checkbox:SetCheckTextureOffsetVal(0,0);
    -- Check if city is in Groups expanded list
    local cityEntry = {
            OwnerID = cityOwnerID,
            CityID = cityID
        };
    local cityIndex = findIndex(m_GroupsFullyExpanded, cityEntry,
CompareCityEntries)
    -- Remove from fully expanded
    if cityIndex > 0 then
        table.remove(m_GroupsFullyExpanded, cityIndex);
    end
    -- Add city to Groups collapsed list, if it does not exist
    cityIndex = findIndex(m_GroupsFullyCollapsed, cityEntry, CompareCityEntries)
    if cityIndex == -1 then
```

```
        table.insert(m_GroupsFullyCollapsed, cityEntry);
    end
    Refresh();
end
function CompareCityEntries( cityEntry1:table, cityEntry2:table )
    if (cityEntry1.OwnerID == cityEntry2.OwnerID) then
        if (cityEntry1.CityID == cityEntry2.CityID) then
            return true;
        end
    end
    return false;
end
function GetCityEntryString( cityEntry:table )
    local pPlayer:table = Players[cityEntry.OwnerID];
    local pCity:table = pPlayer:GetCities():FindID(cityEntry.CityID);
    return Locale.Lookup(pCity:GetName());
end
-- ===========================================================================
--  Trade Route Tracker
-- ===========================================================================
-- ---------------------------------------------------------------------------
-- Trader Route history tracker
-- ---------------------------------------------------------------------------
function UpdateRouteHistoryForTrader(routeInfo:table, routesTable:table)
    if routeInfo.TraderUnitID ~= nil then
        if dbug_print then
            print("Updating trader " .. routeInfo.TraderUnitID .. " with route
history: " .. GetTradeRouteString(routeInfo));
        end
        routesTable[routeInfo.TraderUnitID] = routeInfo;
    else
        if dbug_print then
            print("Could not find the trader unit")
        end
    end
end
-- ===========================================================================
--  Group By Pulldown functions
-- ===========================================================================
function RefreshGroupByPulldown()
    -- Clear current group by entries
    Controls.OverviewGroupByPulldown:ClearEntries();
    m_groupByList = {};
```

```
    -- Build entries
    AddGroupByEntry(Locale.Lookup("LOC_CITY_STATES_NONE"),
GROUP_BY_SETTINGS.NONE);
    AddGroupByEntry(Locale.Lookup("LOC_TRADE_OVERVIEW_ORIGIN"),
GROUP_BY_SETTINGS.ORIGIN);
    AddGroupByEntry(Locale.Lookup("LOC_TRADE_OVERVIEW_DESTINATION"),
GROUP_BY_SETTINGS.DESTINATION);
    AddGroupByEntry(Locale.Lookup("LOC_TRADE_OVERVIEW_ORIGIN_AZ"),
GROUP_BY_SETTINGS.ORIGIN_AZ);
    AddGroupByEntry(Locale.Lookup("LOC_TRADE_OVERVIEW_ORIGIN_ZA"),
GROUP_BY_SETTINGS.ORIGIN_ZA);
    AddGroupByEntry(Locale.Lookup("LOC_TRADE_OVERVIEW_DESTINATION_AZ"),
GROUP_BY_SETTINGS.DESTINATION_AZ);
    AddGroupByEntry(Locale.Lookup("LOC_TRADE_OVERVIEW_DESTINATION_ZA"),
GROUP_BY_SETTINGS.DESTINATION_ZA);
    -- Calculate Internals
    Controls.OverviewGroupByPulldown:CalculateInternals();
    Controls.OverviewGroupByButton:SetText(m_groupByList[m_groupBySelected].grou
pByString);
    UpdateGroupByArrow();
end
function AddGroupByEntry( text:string, id:number )
    local entry:table = {
        groupByString = text,
        groupByID = id
    };
    m_groupByList[id] = entry;
    AddPulldownEntry(text, id);
end
function AddPulldownEntry( pulldownText:string, index:number )
    local groupByPulldownEntry:table = {};
    Controls.OverviewGroupByPulldown:BuildEntry( "OverviewGroupByEntry",
groupByPulldownEntry );
    groupByPulldownEntry.Button:SetText(pulldownText);
    groupByPulldownEntry.Button:SetVoids(i, index);
end
function UpdateGroupByArrow()
    if Controls.OverviewGroupByPulldown:IsOpen() then
        Controls.OverviewGroupByPulldownOpenedArrow:SetHide(true);
        Controls.OverviewGroupByPulldownClosedArrow:SetHide(false);
    else
        Controls.OverviewGroupByPulldownOpenedArrow:SetHide(false);
        Controls.OverviewGroupByPulldownClosedArrow:SetHide(true);
```

```
    end
end
-- Helper method to check if the group setting selected is none
function GroupSettingIsNone(groupSetting)
    if groupSetting == GROUP_BY_SETTINGS.NONE
            or groupSetting == GROUP_BY_SETTINGS.ORIGIN_AZ
            or groupSetting == GROUP_BY_SETTINGS.ORIGIN_ZA
            or groupSetting == GROUP_BY_SETTINGS.DESTINATION_AZ
            or groupSetting == GROUP_BY_SETTINGS.DESTINATION_ZA then
        return true
    end
    return false
end
-- ===========================================================================
--  Filter, Filter Pulldown functions
-- ===========================================================================
function FilterTradeRoutes ( tradeRoutes:table )
    if dbug_print then
        -- print("Current filter: " ..
m_filterList[m_filterSelected].FilterText);
    end
    if m_filterSelected == 1 then
        return tradeRoutes;
    end
    local filtertedRoutes:table = {};
    local hasEntry:boolean = false
    for _, tradeRoute in ipairs(tradeRoutes) do
        local pPlayer = Players[tradeRoute.DestinationCityPlayer];
        if m_filterList[m_filterSelected].FilterFunction and
m_filterList[m_filterSelected].FilterFunction(pPlayer) then
            table.insert(filtertedRoutes, tradeRoute);
            hasEntry = true
        end
    end
    if hasEntry then
        return filtertedRoutes;
    else
        return nil
    end
end
-- ---------------------------------------------------------------------------
-- Filter pulldown functions
-- ---------------------------------------------------------------------------
```

```
function RefreshFilters()
    -- Clear current filters
    Controls.OverviewDestinationFilterPulldown:ClearEntries();
    m_filterList = {};
    m_filterCount = 0;
    -- Add "All" Filter
    AddFilter(Locale.Lookup("LOC_ROUTECHOOSER_FILTER_ALL"), function(a) return
true; end);
    -- Add "International Routes" Filter
    AddFilter(Locale.Lookup("LOC_TRADE_FILTER_INTERNATIONAL_ROUTES_TEXT") ,
IsOtherCiv);
    -- Add "City States with Trade Quest" Filter
    AddFilter(Locale.Lookup("LOC_TRADE_FILTER_CS_WITH_QUEST_TOOLTIP"),
IsCityStateWithTradeQuest);
    -- Add Local Player Filter
    local localPlayerConfig:table = PlayerConfigurations[Game.GetLocalPlayer()];
    local localPlayerName = Locale.Lookup(GameInfo.Civilizations[localPlayerConf
ig:GetCivilizationTypeID()].Name);
    AddFilter(localPlayerName, function(a) return a:GetID() ==
Game.GetLocalPlayer(); end);
    -- Add Filters by Civ
    local players:table = Game.GetPlayers();
    for _, pPlayer in ipairs(players) do
        if pPlayer and pPlayer:IsAlive() and pPlayer:IsMajor() then
            -- Has the local player met the civ?
            if pPlayer:GetDiplomacy():HasMet(Game.GetLocalPlayer()) then
                local playerConfig:table =
PlayerConfigurations[pPlayer:GetID()];
                local name = Locale.Lookup(GameInfo.Civilizations[playerConfig:G
etCivilizationTypeID()].Name);
                AddFilter(name, function(a) return a:GetID() == pPlayer:GetID()
end);
            end
        end
    end
    -- Add "City States" Filter
    AddFilter(Locale.Lookup("LOC_HUD_REPORTS_CITY_STATE"), IsCityState);
    -- Add filters to pulldown
    for filter in pairs(m_filterList) do
        AddFilterEntry(filter);
    end
    -- Select first filter
    Controls.OverviewFilterButton:SetText(m_filterList[m_filterSelected].FilterT
```

```
ext);
    -- Calculate Internals
    Controls.OverviewDestinationFilterPulldown:CalculateInternals();
    UpdateFilterArrow();
end
function AddFilter( filterName:string, filterFunction )
    -- Make sure we don't add duplicate filters
    for _, filter in ipairs(m_filterList) do
        if filter.FilterText == filterName then
            return;
        end
    end
    m_filterCount = m_filterCount + 1;
    m_filterList[m_filterCount] = {FilterText=filterName,
FilterFunction=filterFunction};
end
function AddFilterEntry( filterIndex:number )
    local filterEntry:table = {};
    Controls.OverviewDestinationFilterPulldown:BuildEntry(
"OverviewFilterEntry", filterEntry );
    filterEntry.Button:SetText(m_filterList[filterIndex].FilterText);
    filterEntry.Button:SetVoids(i, filterIndex);
end
function UpdateFilterArrow()
    if Controls.OverviewDestinationFilterPulldown:IsOpen() then
        Controls.OverviewFilterPulldownOpenedArrow:SetHide(true);
        Controls.OverviewFilterPulldownClosedArrow:SetHide(false);
    else
        Controls.OverviewFilterPulldownOpenedArrow:SetHide(false);
        Controls.OverviewFilterPulldownClosedArrow:SetHide(true);
    end
end
-- ===========================================================================
--  Grouped Routes Function
-- ===========================================================================
-- Returns the grouped routes version based on the passed group setting
function GroupRoutes( routesTable, groupSetting )
    if dbug_print then
        print("Group setting: " ..
m_groupByList[m_groupBySelected].groupByString);
    end
    if GroupSettingIsNone(groupSetting) then
        return routesTable
```

```
    end
    local returnRoutesTable:table = {}
    local groupCount:number = 1
    local groupKey:table = {}
    for i=1, #routesTable do
        -- Cant use contor key here since we DONT want a unique key for every
route
        local key:string;
        if groupSetting == GROUP_BY_SETTINGS.ORIGIN then
            key = tostring(routesTable[i].OriginCityPlayer) .. "_" ..
tostring(routesTable[i].OriginCityID)
        elseif groupSetting == GROUP_BY_SETTINGS.DESTINATION then
            key = tostring(routesTable[i].DestinationCityPlayer) .. "_" ..
tostring(routesTable[i].DestinationCityID)
        else
            if dbug_print then
                print("Error: Unknown group setting.")
            end
            return routesTable;
        end
        local index = groupCount;
        if groupKey[key] == nil then
            groupKey[key] = groupCount
            groupCount = groupCount + 1;
        else
            index = groupKey[key]
        end
        if returnRoutesTable[index] == nil then
            returnRoutesTable[index] = {}
        end
        if dbug_print then
            -- print("Inserting " .. GetTradeRouteString(route) .. " in " ..
index)
        end
        returnRoutesTable[index][#(returnRoutesTable[index]) + 1] =
routesTable[i]
    end
    return returnRoutesTable;
end
-- Gets top route from each group and sorts them based on that
function SortGroupedRoutes( groupedRoutes:table, sortSettings:table,
sortSettingsChanged:boolean )
    if (sortSettingsChanged ~= nil and (not sortSettingsChanged)) then
```

```
        if opt_print then
            print("OPT: Not sorting groups")
        end
        return groupedRoutes
    end
    -- Get scores for the top routes, sort them
    local routeScores = {}
    for index=1, #groupedRoutes do
        routeScores[index] = { id = index, score =
ScoreRoute(groupedRoutes[index][1], sortSettings)}
    end
    table.sort(routeScores, function(a, b) return ScoreComp(a, b, sortSettings)
end )
    -- Build new table based on these sorted scores
    local routes = {}
    for i, scoreInfo in ipairs(routeScores) do
        routes[i] = groupedRoutes[scoreInfo.id]
    end
    return routes
    -- if #sortSettings > 0 then
    --     table.sort(groupedRoutes, CompareGroups)
    -- end
end
-- ===========================================================================
--  Sort bar functions
-- ===========================================================================
-- Hides all the ascending/descending arrows
function ResetSortBar()
    Controls.FoodDescArrow:SetHide(true);
    Controls.ProductionDescArrow:SetHide(true);
    Controls.GoldDescArrow:SetHide(true);
    Controls.ScienceDescArrow:SetHide(true);
    Controls.CultureDescArrow:SetHide(true);
    Controls.FaithDescArrow:SetHide(true);
    Controls.TurnsToCompleteDescArrow:SetHide(true);
    Controls.FoodAscArrow:SetHide(true);
    Controls.ProductionAscArrow:SetHide(true);
    Controls.GoldAscArrow:SetHide(true);
    Controls.ScienceAscArrow:SetHide(true);
    Controls.CultureAscArrow:SetHide(true);
    Controls.FaithAscArrow:SetHide(true);
    Controls.TurnsToCompleteAscArrow:SetHide(true);
end
```

```
function RefreshSortBar()
    if m_ctrlDown then
        RefreshSortButtons( m_InGroupSortBySettings );
    else
        RefreshSortButtons( m_GroupSortBySettings );
    end
    if showSortOrdersPermanently or m_shiftDown then
        -- Hide the order texts
        HideSortOrderLabels();
        -- Show them based on current settings
        ShowSortOrderLabels();
    end
end
function ShowSortOrderLabels()
    -- Refresh and show sort orders
    if m_ctrlDown then
        RefreshSortOrderLabels( m_InGroupSortBySettings );
    else
        RefreshSortOrderLabels( m_GroupSortBySettings );
    end
end
function HideSortOrderLabels()
    Controls.FoodSortOrder:SetHide(true);
    Controls.ProductionSortOrder:SetHide(true);
    Controls.GoldSortOrder:SetHide(true);
    Controls.ScienceSortOrder:SetHide(true);
    Controls.CultureSortOrder:SetHide(true);
    Controls.FaithSortOrder:SetHide(true);
    Controls.TurnsToCompleteSortOrder:SetHide(true);
end
-- Shows and hides arrows based on the passed sort order
function SetSortArrow( ascArrow:table, descArrow:table, sortOrder:number )
    if sortOrder == SORT_ASCENDING then
        descArrow:SetHide(true);
        ascArrow:SetHide(false);
    else
        descArrow:SetHide(false);
        ascArrow:SetHide(true);
    end
end
function RefreshSortButtons( sortSettings:table )
    -- Hide all arrows
    ResetSortBar();
```

```
    -- Set disabled color
    Controls.FoodSortButton:SetColorByName("ButtonDisabledCS");
    Controls.ProductionSortButton:SetColorByName("ButtonDisabledCS");
    Controls.GoldSortButton:SetColorByName("ButtonDisabledCS");
    Controls.ScienceSortButton:SetColorByName("ButtonDisabledCS");
    Controls.CultureSortButton:SetColorByName("ButtonDisabledCS");
    Controls.FaithSortButton:SetColorByName("ButtonDisabledCS");
    Controls.TurnsToCompleteSortButton:SetColorByName("ButtonDisabledCS");
    -- Go through settings and display arrows
    for _, sortEntry in ipairs(sortSettings) do
        if sortEntry.SortByID == SORT_BY_ID.FOOD then
            SetSortArrow(Controls.FoodAscArrow, Controls.FoodDescArrow,
sortEntry.SortOrder)
            Controls.FoodSortButton:SetColorByName("ButtonCS");
        elseif sortEntry.SortByID == SORT_BY_ID.PRODUCTION then
            SetSortArrow(Controls.ProductionAscArrow,
Controls.ProductionDescArrow, sortEntry.SortOrder)
            Controls.ProductionSortButton:SetColorByName("ButtonCS");
        elseif sortEntry.SortByID == SORT_BY_ID.GOLD then
            SetSortArrow(Controls.GoldAscArrow, Controls.GoldDescArrow,
sortEntry.SortOrder)
            Controls.GoldSortButton:SetColorByName("ButtonCS");
        elseif sortEntry.SortByID == SORT_BY_ID.SCIENCE then
            SetSortArrow(Controls.ScienceAscArrow, Controls.ScienceDescArrow,
sortEntry.SortOrder)
            Controls.ScienceSortButton:SetColorByName("ButtonCS");
        elseif sortEntry.SortByID == SORT_BY_ID.CULTURE then
            SetSortArrow(Controls.CultureAscArrow, Controls.CultureDescArrow,
sortEntry.SortOrder)
            Controls.CultureSortButton:SetColorByName("ButtonCS");
        elseif sortEntry.SortByID == SORT_BY_ID.FAITH then
            SetSortArrow(Controls.FaithAscArrow, Controls.FaithDescArrow,
sortEntry.SortOrder)
            Controls.FaithSortButton:SetColorByName("ButtonCS");
        elseif sortEntry.SortByID == SORT_BY_ID.TURNS_TO_COMPLETE then
            SetSortArrow(Controls.TurnsToCompleteAscArrow,
Controls.TurnsToCompleteDescArrow, sortEntry.SortOrder)
            Controls.TurnsToCompleteSortButton:SetColorByName("ButtonCS");
        end
    end
end
function RefreshSortOrderLabels( sortSettings:table )
    for _, sortEntry in ipairs(sortSettings) do
```

```
        if sortEntry.SortByID == SORT_BY_ID.FOOD then
            Controls.FoodSortOrder:SetHide(false);
            Controls.FoodSortOrder:SetText(index);
            Controls.FoodSortOrder:SetColorByName("ResFoodLabelCS");
        elseif sortEntry.SortByID == SORT_BY_ID.PRODUCTION then
            Controls.ProductionSortOrder:SetHide(false);
            Controls.ProductionSortOrder:SetText(index);
            Controls.ProductionSortOrder:SetColorByName("ResProductionLabelCS");
        elseif sortEntry.SortByID == SORT_BY_ID.GOLD then
            Controls.GoldSortOrder:SetHide(false);
            Controls.GoldSortOrder:SetText(index);
            Controls.GoldSortOrder:SetColorByName("ResGoldLabelCS");
        elseif sortEntry.SortByID == SORT_BY_ID.SCIENCE then
            Controls.ScienceSortOrder:SetHide(false);
            Controls.ScienceSortOrder:SetText(index);
            Controls.ScienceSortOrder:SetColorByName("ResScienceLabelCS");
        elseif sortEntry.SortByID == SORT_BY_ID.CULTURE then
            Controls.CultureSortOrder:SetHide(false);
            Controls.CultureSortOrder:SetText(index);
            Controls.CultureSortOrder:SetColorByName("ResCultureLabelCS");
        elseif sortEntry.SortByID == SORT_BY_ID.FAITH then
            Controls.FaithSortOrder:SetHide(false);
            Controls.FaithSortOrder:SetText(index);
            Controls.FaithSortOrder:SetColorByName("ResFaithLabelCS");
        elseif sortEntry.SortByID == SORT_BY_ID.TURNS_TO_COMPLETE then
            Controls.TurnsToCompleteSortOrder:SetHide(false);
            Controls.TurnsToCompleteSortOrder:SetText(index);
        end
    end
end
-- ===========================================================================
--  Applicaton level functions
-- ===========================================================================
function Open()
    -- dont show panel if there is no local player
    local localPlayerID = Game.GetLocalPlayer();
    if (localPlayerID == -1) then
        return
    end
    m_AnimSupport.Show();
    UI.PlaySound("CityStates_Panel_Open");
    Refresh();
end
```

```
function Close()
    if not ContextPtr:IsHidden() then
        UI.PlaySound("CityStates_Panel_Close");
    end
    m_AnimSupport.Hide();
    -- Reset sort settings
    m_InGroupSortBySettings = {};
    m_GroupSortBySettings = {};
    -- Reset tab
    m_currentTab = TRADE_TABS.MY_ROUTES;
    -- Reset filter
    m_filterSelected = 1;
end
-- ===========================================================================
--  General helper functions
-- ===========================================================================
function SelectUnit( unit:table )
    local localPlayer = Game.GetLocalPlayer();
    if UI.GetHeadSelectedUnit() ~= unit and localPlayer ~= -1 and localPlayer ==
unit:GetOwner() then
        UI.DeselectAllUnits();
        UI.DeselectAllCities();
        UI.SelectUnit( unit );
    end
    UI.LookAtPlotScreenPosition( unit:GetX(), unit:GetY(), 0.42, 0.5 );
end
function SelectFreeTrader( unit:table, destinationCityOwnerID:number,
destinationCityID:number )
    local localPlayer = Game.GetLocalPlayer();
    if localPlayer == -1 or localPlayer ~= unit:GetOwner() then
        return
    end
    local selectedUnit:table = UI.GetHeadSelectedUnit();
    if selectedUnit == nil or selectedUnit:GetID() ~= unit:GetID() then
        UI.DeselectAllUnits();
        UI.DeselectAllCities();
        -- Don't open screen on unit selection
        LuaEvents.TradeRouteChooser_SkipOpen()
        UI.SelectUnit( unit );
        -- Open screen after new destination info is passed
        LuaEvents.TradeOverview_SelectRouteFromOverview(destinationCityOwnerID,
destinationCityID)
    else
```

```
        LuaEvents.TradeOverview_SelectRouteFromOverview(destinationCityOwnerID,
destinationCityID)
    end
end
function CycleTraders(co)
    if CountTraders(m_AvailableTraders) > 0 then
        coroutine.resume(co)
    else
        if dbug_print then
            print("No Trader available")
        end
    end
end
function TransferTraderTo( unit:table, transferCity:table )
    -- Don't open screen on unit selection
    LuaEvents.TradeRouteChooser_SkipOpen()
    SelectUnit(unit)
    LuaEvents.TradeOverview_ChangeOriginCityFromOverview(transferCity)
end
-- Prevents nill entries being counted as "traders"
function CountTraders( traders )
    local count = 0
    if traders ~= nil then
        for cityID in pairs(traders) do
            if traders[cityID] ~= nil then
                for i in pairs(traders[cityID]) do
                    if traders[cityID][i] ~= nil then
                        count = count + 1
                    end
                end
            end
        end
    end
    return count
end
function RemoveTrader( traderID )
    for cityID in pairs(m_AvailableTraders) do
        for i in pairs(m_AvailableTraders[cityID]) do
            -- Remove trader
            if m_AvailableTraders[cityID][i] == traderID then
                if dbug_print then
                    print("Removing trader " .. traderID .. " from available
traders.")
```

```
                end
                table.remove(m_AvailableTraders[cityID], i)
                -- Check if for that city has no traders. Remove the city entry
if it does
                if table_nnill_count(m_AvailableTraders[cityID]) <= 0 then
                    if dbug_print then
                        print("Removing city " .. cityID)
                    end
                    table.remove(m_AvailableTraders, cityID)
                end
                return -- return here since nothing else is left to do
            end
        end
    end
    if dbug_print then
        print("Could not find trader " .. traderID)
    end
end
-- ===========================================================================
--  Button handler functions
-- ===========================================================================
function OnOpen()
    Open();
end
function OnClose()
    Close();
end
-- ---------------------------------------------------------------------------
-- Tab buttons
-- ---------------------------------------------------------------------------
function OnMyRoutesButton()
    m_currentTab = TRADE_TABS.MY_ROUTES;
    Refresh();
end
function OnRoutesToCitiesButton()
    m_currentTab = TRADE_TABS.ROUTES_TO_CITIES;
    Refresh();
end
function OnAvailableRoutesButton()
    m_currentTab = TRADE_TABS.AVAILABLE_ROUTES;
    Refresh();
end
-- ---------------------------------------------------------------------------
```

```
-- Pulldowns
-- ---------------------------------------------------------------------------
function OnFilterSelected( index:number, filterIndex:number )
    m_filterSelected = filterIndex;
    Controls.OverviewFilterButton:SetText(m_filterList[m_filterSelected].FilterT
ext);
    m_FilterSettingsChanged = true;
    Refresh();
end
function OnGroupBySelected( index:number, groupByIndex:number )
    -- Insert sort entry specific to the group setting
    if GROUP_BY_SETTINGS.ORIGIN_AZ == groupByIndex then
        m_GroupSortBySettings = {}
        InsertSortEntry(SORT_BY_ID.ORIGIN_NAME, SORT_ASCENDING,
m_GroupSortBySettings)
    elseif GROUP_BY_SETTINGS.ORIGIN_ZA == groupByIndex then
        m_GroupSortBySettings = {}
        InsertSortEntry(SORT_BY_ID.ORIGIN_NAME, SORT_DESCENDING,
m_GroupSortBySettings)
    elseif GROUP_BY_SETTINGS.DESTINATION_AZ == groupByIndex then
        m_GroupSortBySettings = {}
        InsertSortEntry(SORT_BY_ID.DESTINATION_NAME, SORT_ASCENDING,
m_GroupSortBySettings)
    elseif GROUP_BY_SETTINGS.DESTINATION_ZA == groupByIndex then
        m_GroupSortBySettings = {}
        InsertSortEntry(SORT_BY_ID.DESTINATION_NAME, SORT_DESCENDING,
m_GroupSortBySettings)
    end
    m_groupBySelected = groupByIndex;
    Controls.OverviewGroupByButton:SetText(m_groupByList[m_groupBySelected].grou
pByString);
    -- Have to rebuild table
    m_GroupSettingsChanged = true;
    Refresh();
end
-- ---------------------------------------------------------------------------
-- Checkbox
-- ---------------------------------------------------------------------------
function OnGroupExpandAll()
    m_GroupExpandAll = false;
    m_GroupCollapseAll = false;
    Controls.GroupCollapseAllCheckBox:SetCheck(false);
    -- Dont do anything, if grouping is none
```

```
    if GroupSettingIsNone(m_groupBySelected) then
        return;
    end
    if Controls.GroupExpandAllCheckBox:IsChecked() then
        m_GroupsFullyCollapsed = {};
        m_GroupExpandAll = true;
    end
    Refresh();
end
function OnGroupCollapseAll()
    m_GroupExpandAll = false;
    m_GroupCollapseAll = false;
    Controls.GroupExpandAllCheckBox:SetCheck(false);
    -- Dont do anything, if grouping is none
    if GroupSettingIsNone(m_groupBySelected) then
        return;
    end
    if Controls.GroupCollapseAllCheckBox:IsChecked() then
        m_GroupsFullyExpanded = {};
        m_GroupCollapseAll = true;
    end
    Refresh();
end
-- ---------------------------------------------------------------------------
-- Sort bar insert buttons
-- ---------------------------------------------------------------------------
-- General method to handle a sort button. Kind of a mess, especially with
handling of different features with key presses. In short:
-- SHIFT    = Clear previous valuse
-- CTRL     = Add to m_InGroupSortBySettings
-- No CTRL  = Add to m_GroupSortBySettings and m_InGroupSortBySettings
-- By default the ascending sort by turns is always added (since these routes
could be hidden in groups)
function OnGeneralSortBy(sortDescArrow, sortByID)
    m_SortSettingsChanged = true;
    -- If shift is not being pressed, reset sort settings
    if not m_shiftDown then
        if not m_ctrlDown then
            m_GroupSortBySettings = {};
        end
        m_InGroupSortBySettings = {};
    end
    -- Remove sort by turns ascending to be added later
```

```
    RemoveSortEntry(SORT_BY_ID.TURNS_TO_COMPLETE, m_InGroupSortBySettings);
    -- Sort based on currently showing icon toggled
    if sortDescArrow:IsHidden() then
        if not m_ctrlDown then
            InsertSortEntry(sortByID, SORT_DESCENDING, m_GroupSortBySettings);
        end
        InsertSortEntry(sortByID, SORT_DESCENDING, m_InGroupSortBySettings);
    else
        if not m_ctrlDown then
            InsertSortEntry(sortByID, SORT_ASCENDING, m_GroupSortBySettings);
        end
        InsertSortEntry(sortByID, SORT_ASCENDING, m_InGroupSortBySettings);
    end
    InsertSortEntry(SORT_BY_ID.TURNS_TO_COMPLETE, SORT_ASCENDING,
m_InGroupSortBySettings);
    RefreshSortBar();
    -- OPT: Dont call refresh while shift is held
    if not m_shiftDown then
        Refresh();
    else
        m_sortCallRefresh = true;
    end
end
function OnSortByFood()
    OnGeneralSortBy(Controls.FoodDescArrow, SORT_BY_ID.FOOD)
end
function OnSortByProduction()
    OnGeneralSortBy(Controls.ProductionDescArrow, SORT_BY_ID.PRODUCTION)
end
function OnSortByGold()
    OnGeneralSortBy(Controls.GoldDescArrow, SORT_BY_ID.GOLD)
end
function OnSortByScience()
    OnGeneralSortBy(Controls.ScienceDescArrow, SORT_BY_ID.SCIENCE)
end
function OnSortByCulture()
    OnGeneralSortBy(Controls.CultureDescArrow, SORT_BY_ID.CULTURE)
end
function OnSortByFaith()
    OnGeneralSortBy(Controls.FaithDescArrow, SORT_BY_ID.FAITH)
end
function OnSortByTurnsToComplete()
    OnGeneralSortBy(Controls.TurnsToCompleteDescArrow,
```

```
SORT_BY_ID.TURNS_TO_COMPLETE)
end
-- ---------------------------------------------------------------------------
-- Sort bar delete buttons
-- ---------------------------------------------------------------------------
-- General method to remove sort button.
-- CTRL     = Remove from m_InGroupSortBySettings
-- No CTRL  = Remove from m_GroupSortBySettings and m_InGroupSortBySettings
function OnGeneralNotSortBy(sortByID)
    m_SortSettingsChanged = true;
    if not m_ctrlDown then
        RemoveSortEntry( sortByID, m_GroupSortBySettings);
    end
    RemoveSortEntry( sortByID, m_InGroupSortBySettings);
    RefreshSortBar();
    -- OPT: Dont call refresh while shift is held
    if not m_shiftDown then
        Refresh();
    else
        m_sortCallRefresh = true;
    end
end
function OnNotSortByFood()
    OnGeneralNotSortBy(SORT_BY_ID.FOOD)
end
function OnNotSortByProduction()
    OnGeneralNotSortBy(SORT_BY_ID.PRODUCTION)
end
function OnNotSortByGold()
    OnGeneralNotSortBy(SORT_BY_ID.GOLD)
end
function OnNotSortByScience()
    OnGeneralNotSortBy(SORT_BY_ID.SCIENCE)
end
function OnNotSortByCulture()
    OnGeneralNotSortBy(SORT_BY_ID.CULTURE)
end
function OnNotSortByFaith()
    OnGeneralNotSortBy(SORT_BY_ID.FAITH)
end
function OnNotSortByTurnsToComplete()
    OnGeneralNotSortBy(SORT_BY_ID.TURNS_TO_COMPLETE)
end
```

```
-- ===========================================================================
--  LUA Event
--  Explicit close (from partial screen hooks), part of closing everything,
-- ===========================================================================
function OnCloseAllExcept( contextToStayOpen:string )
    if contextToStayOpen == ContextPtr:GetID() then return; end
    Close();
end
-- ===========================================================================
--  Game Event
-- ===========================================================================
--  City was selected so close route chooser
function OnCitySelectionChanged(owner, ID, i, j, k, bSelected, bEditable)
    if not ContextPtr:IsHidden() and owner == Game.GetLocalPlayer() then
        OnClose();
    end
end
function OnInterfaceModeChanged( eOldMode:number, eNewMode:number )
    if eNewMode == InterfaceModeTypes.VIEW_MODAL_LENS then
        Close();
    end
end
function OnLocalPlayerTurnEnd()
    if(GameConfiguration.IsHotseat()) then
        Close();
    end
    m_HasBuiltTradeRouteTable = false;
    -- Clear cache and tables to keep memory used low
    CacheEmpty();
    m_AvailableTradeRoutes = nil;
    m_FinalTradeRoutes = nil;
    m_AvailableGroupedRoutes = nil;
end
function OnUnitOperationStarted( ownerID:number, unitID:number,
operationID:number )
    -- Don't do anything for non local players
    if ownerID ~= Game.GetLocalPlayer() then return end
    if m_HasBuiltTradeRouteTable then
        -- Remove unit from available traders
        RemoveTrader(unitID)
        local foundRoute:boolean = false
        if operationID == UnitOperationTypes.MAKE_TRADE_ROUTE then
            -- Unit was just started a trade route. Find the route, and update
```

```
the tables
            local localPlayerCities:table = Players[ownerID]:GetCities();
            for _, city in localPlayerCities:Members() do
                local outgoingRoutes = city:GetTrade():GetOutgoingRoutes();
                for _, route in ipairs(outgoingRoutes) do
                    if route.TraderUnitID == unitID then
                        if dbug_print then
                            print("Found route...")
                        end
                        -- Remove it from the available routes
                        RemoveRouteFromTable(route, m_AvailableGroupedRoutes,
not GroupSettingIsNone(m_groupBySelected));
                        foundRoute = true
                        break
                    end
                end
            end
            if not foundRoute then
                if dbug_print then
                    print("Route not found!!")
                end
                return
            end
            -- Dont refresh, if the window is hidden
            if not ContextPtr:IsHidden() then
                Refresh();
            end
        end
    end
end
-- ===========================================================================
--  UI EVENTS
-- ===========================================================================
function OnInit( isReload:boolean )
    if isReload then
        LuaEvents.GameDebug_GetValues(RELOAD_CACHE_ID);
    end
end
function OnShutdown()
    LuaEvents.GameDebug_AddValue(RELOAD_CACHE_ID, "isHidden",
ContextPtr:IsHidden());
    LuaEvents.GameDebug_AddValue(RELOAD_CACHE_ID, "currentTab", m_currentTab);
    LuaEvents.GameDebug_AddValue(RELOAD_CACHE_ID, "filterSelected",
```

```
m_filterSelected);
    LuaEvents.GameDebug_AddValue(RELOAD_CACHE_ID, "groupBySelected",
m_groupBySelected);
end
-- ---------------------------------------------------------------------------
-- Input handlers.
-- ---------------------------------------------------------------------------
function KeyDownHandler( key:number )
    if key == Keys.VK_SHIFT then
        m_shiftDown = true;
        if not showSortOrdersPermanently then
            ShowSortOrderLabels();
        end
        -- let it fall through
    end
    if key == Keys.VK_CONTROL then
        m_ctrlDown = true;
        RefreshSortBar();
    end
    return false;
end
function KeyUpHandler( key:number )
    if key == Keys.VK_SHIFT then
        m_shiftDown = false;
        if m_sortCallRefresh then
            Refresh();
            m_sortCallRefresh = false;
        end
        if not showSortOrdersPermanently then
            HideSortOrderLabels();
        end
        -- let it fall through
    end
    if key == Keys.VK_CONTROL then
        m_ctrlDown = false;
        RefreshSortBar();
    end
    if key == Keys.VK_ESCAPE then
        Close();
        return true;
    end
    if key == Keys.VK_RETURN then
        -- Don't let enter propigate or it will hit action panel which will
```

```
raise a screen (potentially this one again) tied to the action.
        return true;
    end
    return false;
end
function OnInputHandler( pInputStruct:table )
    -- Call the animation input handler
    -- m_AnimSupport.OnInputHandler ( pInputStruct );
    local uiMsg = pInputStruct:GetMessageType();
    if uiMsg == KeyEvents.KeyDown then return KeyDownHandler(
pInputStruct:GetKey() ); end
    if uiMsg == KeyEvents.KeyUp then return KeyUpHandler( pInputStruct:GetKey()
); end
    return false;
end
-- ===========================================================================
--  LUA EVENT
--  Reload support
-- ===========================================================================
function OnGameDebugReturn( context:string, contextTable:table )
    if context == RELOAD_CACHE_ID then
        if contextTable["isHidden"] ~= nil and not contextTable["isHidden"] then
            Open();
        end
        -- TODO: Add reload support for sort bar
        if contextTable["filterSelected"] ~= nil then
            m_filterSelected = contextTable["filterSelected"];
            Refresh();
        end
        if contextTable["currentTab"] ~= nil then
            m_currentTab = contextTable["currentTab"];
            Refresh();
        end
        if contextTable["groupBySelected"] ~= nil then
            m_groupBySelected = contextTable["groupBySelected"];
            -- Have to rebuild table
            m_HasBuiltTradeRouteTable = false;
            Refresh();
        end
    end
end
function OnPolicyChanged( ePlayer )
    if m_AnimSupport.IsVisible() and ePlayer == Game.GetLocalPlayer() then
```

```
        Refresh();
    end
end
-- ===========================================================================
--  Setup
-- ===========================================================================
function InitButton(control, callbackLClick, callbackRClick)
    control:RegisterCallback(Mouse.eLClick, callbackLClick)
    if callbackRClick ~= nil then
        control:RegisterCallback(Mouse.eRClick, callbackRClick)
    end
    control:RegisterCallback( Mouse.eMouseEnter, function()
UI.PlaySound("Main_Menu_Mouse_Over") end)
end
function Initialize()
    if dbug_print then
        print("Initializing BTS Trade Overview");
    end
    -- Initialize tracker
    TradeSupportTracker_Initialize();
    -- Input handler
    ContextPtr:SetInputHandler( OnInputHandler, true );
    -- Control Events
    InitButton(Controls.CloseButton, OnClose)
    InitButton(Controls.MyRoutesButton, OnMyRoutesButton)
    InitButton(Controls.RoutesToCitiesButton, OnRoutesToCitiesButton)
    InitButton(Controls.AvailableRoutesButton, OnAvailableRoutesButton)
    -- Control events - sort bar
    InitButton(Controls.FoodSortButton, OnSortByFood, OnNotSortByFood)
    InitButton(Controls.ProductionSortButton, OnSortByProduction,
OnNotSortByProduction)
    InitButton(Controls.GoldSortButton, OnSortByGold, OnNotSortByGold)
    InitButton(Controls.ScienceSortButton, OnSortByScience, OnNotSortByScience)
    InitButton(Controls.CultureSortButton, OnSortByCulture, OnNotSortByCulture)
    InitButton(Controls.FaithSortButton, OnSortByFaith, OnNotSortByFaith)
    InitButton(Controls.TurnsToCompleteSortButton, OnSortByTurnsToComplete,
OnNotSortByTurnsToComplete)
    --Filter Pulldown
    Controls.OverviewFilterButton:RegisterCallback( Mouse.eLClick,
UpdateFilterArrow );
    Controls.OverviewDestinationFilterPulldown:RegisterSelectionCallback(
OnFilterSelected );
    -- Group By Pulldown
```

```
    Controls.OverviewGroupByButton:RegisterCallback( Mouse.eLClick,
UpdateGroupByArrow );
    Controls.OverviewGroupByPulldown:RegisterSelectionCallback(
OnGroupBySelected );
    InitButton(Controls.GroupExpandAllCheckBox, OnGroupExpandAll)
    InitButton(Controls.GroupCollapseAllCheckBox, OnGroupCollapseAll)
    -- Lua Events
    LuaEvents.PartialScreenHooks_OpenTradeOverview.Add( OnOpen );
    LuaEvents.PartialScreenHooks_CloseTradeOverview.Add( OnClose );
    LuaEvents.PartialScreenHooks_CloseAllExcept.Add( OnCloseAllExcept );
    -- Animation Controller
    m_AnimSupport = CreateScreenAnimation(Controls.SlideAnim);
    -- Rundown / Screen Events
    Events.SystemUpdateUI.Add(m_AnimSupport.OnUpdateUI);
    Controls.Title:SetText(Locale.Lookup("LOC_TRADE_OVERVIEW_TITLE"));
    -- Game Engine Events
    Events.CitySelectionChanged.Add( OnCitySelectionChanged );
    Events.UnitOperationStarted.Add( OnUnitOperationStarted );
    Events.GovernmentPolicyChanged.Add( OnPolicyChanged );
    Events.GovernmentPolicyObsoleted.Add( OnPolicyChanged );
    Events.LocalPlayerTurnEnd.Add( OnLocalPlayerTurnEnd );
    Events.InterfaceModeChanged.Add( OnInterfaceModeChanged );
    -- Hot-Reload Events
    ContextPtr:SetInitHandler(OnInit);
    ContextPtr:SetShutdown(OnShutdown);
    LuaEvents.GameDebug_Return.Add(OnGameDebugReturn);
end
Initialize();
```

```
<?xml version="1.0" encoding="utf-8"?>
<Context>
  <SlideAnim Style="RundownAnimBG" Speed="5">
    <!-- Title Area -->
    <Grid Anchor="C,T" Size="parent-6,140" Offset="0,50"
Texture="Controls_TitleBarDark" SliceCorner="21,17" SliceTextureSize="42,34">
      <!-- ===================================================================
-->
      <!--  Tab Container
-->
      <!-- ===================================================================
-->
      <Container ID="TabHeader" Anchor="C,T" Offset="0,-10" Size="540,61">
        <Grid Anchor="L,T" Size="parent,61" Texture="Controls_TabLedge2"
SliceCorner="194,18" SliceSize="52,26" SliceTextureSize="438,61">
          <!-- My Routes Button -->
          <GridButton ID="MyRoutesButton" Size="150,34" Offset="30,13"
Style="TabButton">
            <GridButton ID="MyRoutesSelected" Size="parent,parent"
Style="TabButtonSelected" ConsumeMouseButton="0" ConsumeMouseOver="1" />
            <Label ID="MyRoutesTabLabel" Anchor="C,C" Offset="0,1"
Style="TabFont" String="LOC_TRADE_OVERVIEW_MY_ROUTES"/>
            <Image ID="MyRoutesSelectedArrow" Anchor="C,T" Offset="0,0"
Texture="Controls_TabSelectArrow.dds"></Image>
            <Label ID="MyRoutesTabSelectedLabel" Anchor="C,C" Offset="0,1"
Style="TabSelectedFont" String="LOC_TRADE_OVERVIEW_MY_ROUTES" Hidden="1"/>
          </GridButton>
          <!-- Routes to Cities Button  -->
          <GridButton ID="RoutesToCitiesButton" Size="166,34" Offset="188,13"
Style="TabButton">
            <GridButton ID="RoutesToCitiesSelected" Size="parent,parent"
Style="TabButtonSelected" ConsumeMouseButton="0" ConsumeMouseOver="1" />
            <Label ID="RoutesToCitiesTabLabel" Anchor="C,C" Offset="0,1"
Style="TabFont" String="LOC_TRADE_OVERVIEW_ROUTES_TO_MY_CITIES"/>
            <Image ID="RoutesToCitiesSelectedArrow" Anchor="C,T" Offset="0,0"
Texture="Controls_TabSelectArrow.dds"/>
            <Label ID="RoutesToCitiesTabSelectedLabel" Anchor="C,C" Offset="0,1"
Style="TabSelectedFont" String="LOC_TRADE_OVERVIEW_ROUTES_TO_MY_CITIES"
Hidden="1"/>
          </GridButton>
          <!-- Available Routes Button  -->
          <GridButton ID="AvailableRoutesButton" Size="150,34" Offset="360,13"
Style="TabButton">
```

```
            <GridButton ID="AvailableRoutesSelected" Size="parent,parent"
Style="TabButtonSelected" ConsumeMouseButton="0" ConsumeMouseOver="1" />
            <Label ID="AvailableRoutesTabLabel" Anchor="C,C" Offset="0,1"
Style="TabFont" String="LOC_TRADE_OVERVIEW_AVAILABLE_ROUTES"/>
            <Image ID="AvailableRoutesSelectedArrow" Anchor="C,T" Offset="0,0"
Texture="Controls_TabSelectArrow.dds"/>
            <Label ID="AvailableRoutesTabSelectedLabel" Anchor="C,C"
Offset="0,1" Style="TabSelectedFont"
String="LOC_TRADE_OVERVIEW_AVAILABLE_ROUTES" Hidden="1"/>
          </GridButton>
        </Grid>
      </Container>
      <!-- ================================================================
-->
      <!--  Header                                                        -->
      <!-- ================================================================
-->
      <!-- <Label String="BUILDING" Style="FontFlair14" FontStyle="Stroke"
SmallCaps="20" SmallCapsType="EveryWord" Color="249, 249, 249, 255" /> -->
      <Grid ID="HeaderFrame" Anchor="C,T" Offset="0,40" Size="parent-20,42"
Texture="Controls_DecoFrame" SliceCorner="19,18" SliceSize="1,1"
SliceTextureSize="40,38" Color="31,44,53,255">
        <Stack ID="HeaderStack" Anchor="C,C" Offset="0,0" StackGrowth="Right"
StackPadding="6">
          <Label ID="HeaderLabel" Style="TradeOverviewHeader"/>
          <Label ID="ActiveRoutesLabel" Style="TradeOverviewActiveRoutes"/>
        </Stack>
      </Grid>
      <!-- ================================================================
-->
      <!--  Sub Header - Contains Column Names, Filter Pulldown, Sort Bar
-->
      <!-- ================================================================
-->
      <!-- Group by Pulldown -->
      <PullDown ID="OverviewGroupByPulldown" ConsumeMouse="0" Anchor="L,B"
Offset="18,5" Size="155,26" AutoSizePopUp="1" AutoFlip="1"
ScrollThreshold="400">
        <ButtonData>
          <GridButton ID="OverviewGroupByButton"
ToolTip="LOC_TRADE_GROUP_SETTINGS" TextAnchor="C,C" TextOffset="8,0"
Style="FontNormal12" FontStyle="Shadow" EffectColor="0,0,0,255" Offset="0,-25"
Size="parnet,parnet" Texture="Controls_ButtonControl.dds" SliceCorner="10,10"
```

```
SliceSize="1,1" SliceTextureSize="24,24" />
        </ButtonData>
        <GridData InnerPadding="15,15" Offset="0,0" Anchor="L,T"
Style="Drawer"/>
        <ScrollPanelData Anchor="L,C"  Vertical="1" Size="11,14"  Offset="0,0"
AutoScrollBar="1">
          <ScrollBar Style="ScrollVerticalBacking" Anchor="L,T" AnchorSide="I,I"
Color="28,60,90,255" Offset="-2,2">
            <Thumb Style="ScrollThumbAlt" Color="28,60,90,255" />
          </ScrollBar>
        </ScrollPanelData>
        <StackData StackGrowth="Bottom" Offset="0,0" Size="200,400" Anchor="L,T"
/>
        <InstanceData Name="OverviewGroupByEntry">
          <GridButton Anchor="L,T" TextOffset="8,0" ID="Button" Size="265,26"
Offset="1,0" Style="FontNormal12" FontStyle="Shadow" EffectColor="0,0,0,255"
Texture="Controls_ButtonControl.dds" SliceCorner="10,10" SliceSize="1,1"
SliceTextureSize="24,24" StateOffsetIncrement="0,24"/>
        </InstanceData>
        <!-- Show Route Text -->
        <Image ID="OverviewGroupByPulldownOpenedArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,0" Size="20,16"
Anchor="L,T" Offset="6,-20"/>
        <Image ID="OverviewGroupByPulldownClosedArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,60" Size="20,16"
Anchor="L,T" Offset="6,-25"/>
      </PullDown>
      <!-- Filter Pulldown -->
      <PullDown ID="OverviewDestinationFilterPulldown" ConsumeMouse="0"
Anchor="C,B" Offset="30,5" Size="220,26" AutoSizePopUp="1" AutoFlip="1"
ScrollThreshold="400">
        <ButtonData>
          <GridButton ID="OverviewFilterButton"
ToolTip="LOC_TRADE_FILTER_SETTINGS" TextAnchor="C,C" TextOffset="8,0"
Style="FontNormal12" FontStyle="Shadow" EffectColor="0,0,0,255" Offset="0,-25"
Size="parnet,parnet" Texture="Controls_ButtonControl.dds" SliceCorner="10,10"
SliceSize="1,1" SliceTextureSize="24,24" />
        </ButtonData>
        <GridData InnerPadding="15,15" Offset="0,0" Anchor="L,T"
Style="Drawer"/>
        <ScrollPanelData Anchor="L,C"  Vertical="1" Size="11,14"  Offset="0,0"
AutoScrollBar="1">
          <ScrollBar Style="ScrollVerticalBacking" Anchor="L,T" AnchorSide="I,I"
```

```
Color="28,60,90,255" Offset="-2,2">
            <Thumb Style="ScrollThumbAlt" Color="28,60,90,255" />
          </ScrollBar>
        </ScrollPanelData>
        <StackData StackGrowth="Bottom" Offset="0,0" Size="200,400" Anchor="L,T"
/>
        <InstanceData Name="OverviewFilterEntry">
          <GridButton Anchor="L,T" TextOffset="8,0" ID="Button" Size="265,26"
Offset="1,0" Style="FontNormal12" FontStyle="Shadow" EffectColor="0,0,0,255"
Texture="Controls_ButtonControl.dds" SliceCorner="10,10" SliceSize="1,1"
SliceTextureSize="24,24" StateOffsetIncrement="0,24"/>
        </InstanceData>
        <!-- Show Route Text -->
        <Image ID="OverviewFilterPulldownOpenedArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,0" Size="20,16"
Anchor="L,T" Offset="6,-20"/>
        <Image ID="OverviewFilterPulldownClosedArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,60" Size="20,16"
Anchor="L,T" Offset="6,-25"/>
      </PullDown>
      <!-- Checkbox -->
      <Stack StackGrowth="Left" Anchor="R,B" Offset="-17,24" StackPadding="-8">
        <CheckBox ID="GroupExpandAllCheckBox"
String="LOC_TRADE_EXPAND_ALL_BUTTON_TEXT" TextOffset="53,-2"
Style="CityPanelCBFaith" IsChecked="0" />
        <CheckBox ID="GroupCollapseAllCheckBox"
String="LOC_TRADE_COLLAPSE_ALL_BUTTON_TEXT" TextOffset="53,-2"
Style="CityPanelCBFaith" IsChecked="0" />
      </Stack>
      <Container Anchor="C,T" Size="parent-40, 35" Offset="0,88">
        <Grid Style="ColumnHeader" Anchor="C,B" Size="parent+15, 22"
Offset="0,-11">
          <Label String="LOC_TRADE_SORT_BY_TEXT" Anchor="L,C"
Style="FontNormal11" Offset="15,-1"/>
        </Grid>
        <!-- Sort Yield Stack -->
        <Stack Size="50,50" Anchor="L,B" StackGrowth="Right" Offset="82,-9"
StackPadding="2">
          <GridButton ID="FoodSortButton"
ToolTip="LOC_TRADE_SORT_BY_FOOD_TOOLTIP" Size="48,parent-17"
Color="255,255,255,220" Style="PanelButtonLightweight">
            <Label ID="FoodSortOrder" Hidden="1" Anchor="C,C" Offset="1,0"
String="9" Style="FontNormal12"/>
```

```
            <Label ID="FoodSortLabel" Anchor="C,C" Offset="15,1"
Style="FontNormal12" String="[Icon_Food]"/>
            <Image ID="FoodDescArrow" Texture="Controls_ButtonExtendSmall2"
TextureOffset="0,0" Size="20,16" Anchor="C,C" Offset="-12,0"/>
            <Image ID="FoodAscArrow" Texture="Controls_ButtonExtendSmall2"
TextureOffset="0,60" Size="20,16" Anchor="C,C" Offset="-12,-4"/>
          </GridButton>
          <GridButton ID="ProductionSortButton"
ToolTip="LOC_TRADE_SORT_BY_PRODUCTION_TOOLTIP" Size="48,parent-17"
Color="255,255,255,220" Style="PanelButtonLightweight">
            <Label ID="ProductionSortOrder" Hidden="1" Anchor="C,C" Offset="1,0"
String="9" Style="FontNormal12"/>
            <Label ID="ProductionSortLabel" Anchor="C,C" Offset="15,1"
Style="FontNormal12" String="[Icon_Production]"/>
            <Image ID="ProductionDescArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,0" Size="20,16"
Anchor="C,C" Offset="-12,0"/>
            <Image ID="ProductionAscArrow" Texture="Controls_ButtonExtendSmall2"
TextureOffset="0,60" Size="20,16" Anchor="C,C" Offset="-12,-4"/>
          </GridButton>
          <GridButton ID="GoldSortButton"
ToolTip="LOC_TRADE_SORT_BY_GOLD_TOOLTIP" Size="48,parent-17"
Color="255,255,255,220" Style="PanelButtonLightweight">
            <Label ID="GoldSortOrder" Hidden="1" Anchor="C,C" Offset="1,0"
String="9" Style="FontNormal12"/>
            <Label ID="GoldSortLabel" Anchor="C,C" Offset="15,1"
Style="FontNormal12" String="[Icon_Gold]"/>
            <Image ID="GoldDescArrow" Texture="Controls_ButtonExtendSmall2"
TextureOffset="0,0" Size="20,16" Anchor="C,C" Offset="-12,0"/>
            <Image ID="GoldAscArrow" Texture="Controls_ButtonExtendSmall2"
TextureOffset="0,60" Size="20,16" Anchor="C,C" Offset="-12,-4"/>
          </GridButton>
          <GridButton ID="ScienceSortButton"
ToolTip="LOC_TRADE_SORT_BY_SCIENCE_TOOLTIP" Size="48,parent-17"
Color="255,255,255,220" Style="PanelButtonLightweight">
            <Label ID="ScienceSortOrder" Hidden="1" Anchor="C,C" Offset="1,0"
String="9" Style="FontNormal12"/>
            <Label ID="ScienceSortLabel" Anchor="C,C" Offset="15,1"
Style="FontNormal12" String="[Icon_Science]"/>
            <Image ID="ScienceDescArrow" Texture="Controls_ButtonExtendSmall2"
TextureOffset="0,0" Size="20,16" Anchor="C,C" Offset="-12,0"/>
            <Image ID="ScienceAscArrow" Texture="Controls_ButtonExtendSmall2"
TextureOffset="0,60" Size="20,16" Anchor="C,C" Offset="-12,-4"/>
```

```
          </GridButton>
          <GridButton ID="CultureSortButton"
ToolTip="LOC_TRADE_SORT_BY_CULTURE_TOOLTIP" Size="48,parent-17"
Color="255,255,255,220" Style="PanelButtonLightweight">
            <Label ID="CultureSortOrder" Hidden="1" Anchor="C,C" Offset="1,0"
String="9" Style="FontNormal12"/>
            <Label ID="CultureSortLabel" Anchor="C,C" Offset="15,1"
Style="FontNormal12" String="[Icon_Culture]"/>
            <Image ID="CultureDescArrow" Texture="Controls_ButtonExtendSmall2"
TextureOffset="0,0" Size="20,16" Anchor="C,C" Offset="-12,0"/>
            <Image ID="CultureAscArrow" Texture="Controls_ButtonExtendSmall2"
TextureOffset="0,60" Size="20,16" Anchor="C,C" Offset="-12,-4"/>
          </GridButton>
          <GridButton ID="FaithSortButton"
ToolTip="LOC_TRADE_SORT_BY_FAITH_TOOLTIP" Size="48,parent-17"
Color="255,255,255,220" Style="PanelButtonLightweight">
            <Label ID="FaithSortOrder" Hidden="1" Anchor="C,C" Offset="1,0"
String="9" Style="FontNormal12"/>
            <Label ID="FaithSortLabel" Anchor="C,C" Offset="15,0"
Style="FontNormal12" String="[Icon_Faith]"/>
            <Image ID="FaithDescArrow" Texture="Controls_ButtonExtendSmall2"
TextureOffset="0,0" Size="20,16" Anchor="C,C" Offset="-12,0"/>
            <Image ID="FaithAscArrow" Texture="Controls_ButtonExtendSmall2"
TextureOffset="0,60" Size="20,16" Anchor="C,C" Offset="-12,-4"/>
          </GridButton>
          <GridButton ID="TurnsToCompleteSortButton"
ToolTip="LOC_TRADE_SORT_BY_TURNS_REMAINING_TOOLTIP" Size="48,parent-17"
Color="255,255,255,220" Style="PanelButtonLightweight" Offset="36,0">
            <Label ID="TurnsToCompleteSortOrder" Hidden="1" Anchor="C,C"
Offset="1,0" String="9" Style="FontNormal12"/>
            <Label ID="TurnsToCompleteSortLabel" Anchor="C,C" Offset="15,0"
Style="FontNormal12" String="[Icon_Turn]"/>
            <Image ID="TurnsToCompleteDescArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,0" Size="20,16"
Anchor="C,C" Offset="-12,0"/>
            <Image ID="TurnsToCompleteAscArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,60" Size="20,16"
Anchor="C,C" Offset="-12,-4"/>
          </GridButton>
        </Stack>
        <!-- <Box Size="300,1" Anchor="L,B" Offset="56,4" Color="50,53,54,150"/>
-->
      </Container>
```

```
      <!-- ================================================================
-->
      <!--  Close Button                                                  -->
      <!-- ================================================================
-->
      <Button ID="CloseButton" Anchor="R,T" Offset="5,40"
Style="CloseButtonSmall" />
    </Grid>
    <!-- ================================================================ -->
    <!--  Body Area                                                       -->
    <!-- ================================================================ -->
    <Container Anchor="C,B" Size="parent, parent-195" Offset="0,5">
      <ScrollPanel ID="BodyScrollPanel" Size="495,parent" Vertical="1">
        <ScrollBar Anchor="R,C" AnchorSide="O,I"  Offset="0,0"
Style="ScrollVerticalBar"/>
        <!-- Stack -->
        <Stack ID="BodyStack" StackGrowth="Down" StackPadding="4"/>
      </ScrollPanel>
    </Container>
  </SlideAnim>
  <!-- ================================================================ -->
  <!--  Instances                                                       -->
  <!-- ================================================================ -->
  <!-- Section Divider -->
  <Instance Name="SectionDividerInstance">
    <Container ID="Top" Size="parent,25" Anchor="C,C" Offset="8,0">
      <Grid Style="DivHeader" Size="parent,parent" Color="255,255,255,255"
Anchor="C,T" Offset="0,0">
        <!-- <Grid Anchor="C,C" Offset="-9,0" Size="parent-60,8"
Texture="Controls_Div2" SliceCorner="27,4" SliceTextureSize="54,8"
Color="47,63,76,255" /> -->
      </Grid>
    </Container>
  </Instance>
  <!-- Section Header -->
  <Instance Name="HeaderInstance">
    <Container ID="Top" Size="485,42" Offset="10,0">
      <GridButton ID="HeaderGrid" Anchor="C,B" Texture="TradeOverview_Subheader"
SliceCorner="12,11" SliceSize="2,2" Color="67,70,72,255" Size="parent,parent">
        <!-- <Grid           ID="CityBannerFill2"           Anchor="C,C"
Size="Parent-7,Parent-6"  Offset="-1,-1" Texture="Banner_Darker"
SliceTextureSize="196,34" SliceCorner="98,17"/> -->
        <Grid           ID="CityBannerFill"           Anchor="C,C"
```

```
Size="Parent-7,Parent-6"  Offset="-1,-1"  Texture="Banner_Base"
SliceTextureSize="196,34" SliceCorner="98,17"/>
       <!--  <Grid           ID="CityBannerFill3"          Anchor="C,C"
Size="Parent-7,Parent-6"  Offset="-1,-1"  Texture="Banner_Lighter"
SliceTextureSize="196,34" SliceCorner="98,17"/> -->
        <Label ID="HeaderLabel" Anchor="L,C" Offset="36,0" Style="FontFlair16"
SmallCaps="20" SmallCapsType="EveryWord" Color="Beige"/>
        <!-- Expand routes button -->
        <CheckBox ID="RoutesExpand" ButtonSize="41,26"
CheckTexture="Controls_ExpandButton" CheckSize="22,22" CheckTextureOffset="0,0"
UnCheckTexture="Controls_ExpandButton" UnCheckSize="22,22"
UnCheckTextureOffset="0,22" UseSelectedTextures="1" Anchor="R,T" Offset="-7,9"
/>
        <!-- Info Stack -->
        <Stack StackGrowth="Left" Anchor="R,T" Offset="43,4" StackPadding="5">
          <!-- Number of Routes Info Label -->
          <Label ID="RouteCountLabel" String="99 / 99" Anchor="R,T" Offset="5,8"
Style="FontFlair16" FontStyle="shadow" Color="Beige"/>
          <!-- Trading Post Indicator -->
          <Grid ID="TradingPostIndicator" Size="22,22" Align="Center"
Anchor="R,T" Offset="0,4" Color="25,33,35,0">
            <Label ID="TradingPostIcon" String="[Icon_TradingPost]"
Style="FontNormal20"/>
          </Grid>
          <!-- Visibility Bonus -->
          <Grid ID="VisibilityBonusGrid" Size="23,23" Anchor="R,T" Offset="0,4"
Color="25,33,35,0">
            <Button ID="VisibilityBonusIcon" Anchor="C,C" Size="22,22"
Offset="0,0" Texture="Diplomacy_VisibilityIcons" StateOffsetIncrement="22,0"
Disabled="1"/>
          </Grid>
          <!-- Tourism Bonus -->
          <Grid ID="TourismBonusGrid" Anchor="R,T" Size="66,20" Offset="0,5"
Color="25,33,35,0">
            <Image ID="TourismBonusIcon" Anchor="C,C" Size="24,23" Offset="18,1"
Texture="Tourism_VisitingSmallGrey"/>
            <Label ID="TourismBonusPercentage" Anchor="C,C" Align="R,C"
Size="40,24" Offset="-18,1" Style="FontNormal14"
String="LOC_TRADE_OVERVIEW_TOURISM_BONUS" Color="Beige"/>
          </Grid>
        </Stack>
      </Grid>
    </Container>
```

```
  </Instance>
  <!-- Route Entry -->
  <Instance Name="RouteInstance">
    <Container ID="Top" Size="485,78" Offset="10,0">
      <GridButton ID="GridButton" Size="parent,parent" Color="255,255,255,150">
          <GridData Texture="Controls_ListButton" SliceCorner="209,29"
SliceTextureSize="417,51" StateOffsetIncrement="0,102"/>
          <!-- Vertical Divider Background -->
          <Stack Size="50,50"  Anchor="C,B" StackGrowth="Right" Offset="0,4">
            <Box Size="1,parent-34" Anchor="L,B" Color="50,53,54,150"/>
            <Image Size="48,parent-34" Anchor="L,B"
Texture="Espionage_ColumnShading"/>
            <Box Size="1,parent-34" Anchor="L,B" Color="50,53,54,150"/>
            <Box Size="1,parent-34" Anchor="L,B" Color="50,53,54,150"/>
            <Image Size="48,parent-34" Anchor="L,B"
Texture="Espionage_ColumnShading"/>
            <Box Size="1,parent-34" Anchor="L,B" Color="50,53,54,150"/>
            <Box Size="1,parent-34" Anchor="L,B" Color="50,53,54,150"/>
            <Image Size="48,parent-34" Anchor="L,B"
Texture="Espionage_ColumnShading"/>
            <Box Size="1,parent-34" Anchor="L,B" Color="50,53,54,150"/>
            <Box Size="1,parent-34" Anchor="L,B" Color="50,53,54,150"/>
            <Image Size="48,parent-34" Anchor="L,B"
Texture="Espionage_ColumnShading"/>
            <Box Size="1,parent-34" Anchor="L,B" Color="50,53,54,150"/>
            <Box Size="1,parent-34" Anchor="L,B" Color="50,53,54,150"/>
            <Image Size="48,parent-34" Anchor="L,B"
Texture="Espionage_ColumnShading"/>
            <Box Size="1,parent-34" Anchor="L,B" Color="50,53,54,150"/>
            <Box Size="1,parent-34" Anchor="L,B" Color="50,53,54,150"/>
            <Image Size="48,parent-34" Anchor="L,B"
Texture="Espionage_ColumnShading"/>
            <Box Size="1,parent-34" Anchor="L,B" Color="50,53,54,150"/>
          </Stack>
          <!-- Route Status Font Icon -->
          <Label ID="RouteStatusFontIcon" Anchor="L,T" Offset="10,8"
String="[Icon_TradeRouteLarge]" Hidden="1"/>
          <Grid ID="BannerBase" Size="parent,37" Anchor="C,T" Offset="0,1"
Texture="CityPanel_BannerBase"       SliceCorner="20,10" SliceSize="160,1"
SliceTextureSize="199,33" Color="150,170,100,255">
            <Grid             ID="BannerDarker"         Anchor="L,T"
Offset="4,2"     Size="parent-8,parent-10"   Texture="CityPanel_BannerDarker"
SliceCorner="95,10" SliceSize="1,1" SliceTextureSize="191,23" Color="0,0,0,100"
```

```
/>
              <Grid             ID="BannerLighter"        Anchor="L,T"
Offset="4,2"     Size="parent-8,parent-10"   Texture="CityPanel_BannerLighter"
SliceCorner="95,10" SliceSize="1,1" SliceTextureSize="191,23"
Color="255,255,255,255" />
          </Grid>
          <!-- Route Label -->
          <Label ID="RouteLabel" Anchor="L,T" Offset="33,10" Style="FontFlair16"
SmallCaps="20" SmallCapsType="EveryWord" Color="Beige"/>
          <!-- Info Stack -->
          <Stack StackGrowth="Left" Anchor="R,T" Offset="70,0" StackPadding="5">
            <!-- Trading Post Indicator -->
            <Grid ID="TradingPostIndicator" Size="22,22" Anchor="R,T"
Offset="0,4" Color="25,33,35,0">
              <Label ID="TradingPostIcon" String="[Icon_TradingPost]"
Anchor="C,C"/>
            </Grid>
            <!-- Visibility Bonus -->
            <Grid ID="VisibilityBonusGrid" Size="22,22" Anchor="R,T"
Offset="0,4" Color="25,33,35,0">
              <Button ID="VisibilityBonusIcon" Anchor="L,T" Size="22,22"
Offset="0,0" Texture="Diplomacy_VisibilityIcons" StateOffsetIncrement="22,0"
Disabled="1"/>
            </Grid>
            <!-- Tourism Bonus -->
            <Grid ID="TourismBonusGrid" Anchor="R,T" Size="66,20" Offset="0,5"
Color="25,33,35,0">
              <Image ID="TourismBonusIcon" Anchor="C,C" Size="24,20"
Offset="18,0" Texture="Tourism_VisitingSmallGrey"/>
              <Label ID="TourismBonusPercentage" Anchor="C,C" Align="R,C"
Size="40,24" Offset="-18,1" Style="FontNormal14"
String="LOC_TRADE_OVERVIEW_TOURISM_BONUS" Color="Beige"/>
            </Grid>
          </Stack>
          <!-- City State Quest Icon -->
          <Label ID="CityStateQuestIcon" Anchor="R,B" Offset="40,28"
Style="FontNormal16" FontStyle="Shadow" String="[ICON_CityStateQuest]"
Color="255,0,0,255" EffectColor="255,0,0,255" GradientColor="255,0,0,255" />
          <!-- Route Distance -->
          <Stack ID="RouteDistanceStack" Anchor="R,T" Offset="12,4"
StackGrowth="Right" StackPadding="2">
            <Label ID="TurnsToCompleteIcon"  String="[ICON_Turn]" Anchor="C,C"
Offset="0,0"  Style="FontFlair16" FontStyle="Stroke" EffectColor="0,0,0,25" />
```

```
            <Label ID="TurnsToComplete" Offset="0,4" Style="FontNormal16"
String="00" ToolTip="LOC_TRADE_OVERVIEW_TOOLTIP_TOTAL_ROUTE_TURNS"/>
          </Stack>
          <!-- Route Duration -->
          <Label Anchor="R,T" Offset="26,7" Style="FontNormal16"
String="[Icon_Turn]" Hidden="1"/>
          <Label Anchor="R,T" Offset="10,8" Style="FontNormal16" String="00"
Hidden="1"/>
          <!-- Origin Civ Icon -->
          <Image ID="OriginCivIconBacking" Anchor="L,B" Size="30,30"
Offset="8,12" Texture="CircleBacking30" Color="100,100,100,255">
            <Image ID="OriginCivIcon" Anchor="C,C" Size="30,30" Offset="0,0"
Icon="ICON_CIVILIZATION_UNKNOWN" IconSize="30"/>
          </Image>
          <!-- Origin Civ Arrow -->
          <Image ID="OriginCivArrow" Anchor="L,B" Size="20,19" Offset="46,25"
FlipX="1" Texture="TradeOverview_Benefactor"/>
           <!-- Destination Civ Icon -->
           <Image ID="DestinationCivIconBacking" Anchor="R,B" Size="30,30"
Offset="8,12" Texture="CircleBacking30" Color="100,100,100,255">
             <Image ID="DestinationCivIcon" Anchor="C,C" Size="30,30"
Offset="0,0" Icon="ICON_CIVILIZATION_UNKNOWN" IconSize="30"/>
           </Image>
          <!-- Destination Civ Arrow -->
          <Image ID="DestinationCivArrow" Anchor="R,B" Size="20,19"
Offset="45,3" Texture="TradeOverview_Benefactor"/>
          <!-- Origin/Destination Divider Box -->
          <Box Size="300,2" Anchor="C,C" Offset="0,13" Color="50,53,54,255"/>
          <!-- Resource Stack -->
          <Stack ID="ResourceStack" StackGrowth="Down" Anchor="C,T"
Offset="0,31" StackPadding="4">
            <Stack ID="OriginYields" Anchor="C,T" StackGrowth="Right"
StackPadding="2">
              <Container Size="48,20">
                <Label ID="OriginYieldFoodLabel" Color="Food" Anchor="C,C"
Offset="0,0" Style="FontNormal14" FontStyle="Shadow" String="+15[Icon_Food]"/>
              </Container>
              <Container Size="48,20">
                <Label ID="OriginYieldProductionLabel" Color="Production"
Anchor="C,C" Offset="0,0" Style="FontNormal14" FontStyle="Shadow"
String="+15[Icon_Production]"/>
              </Container>
              <Container Size="48,20">
```

```
                <Label ID="OriginYieldGoldLabel" Color="Gold" Anchor="C,C"
Offset="0,0" Style="FontNormal14" FontStyle="Shadow" String="+15[Icon_Gold]"/>
              </Container>
              <Container Size="48,20">
                <Label ID="OriginYieldScienceLabel" Color="Science" Anchor="C,C"
Offset="0,0" Style="FontNormal14" FontStyle="Shadow"
String="+15[Icon_Science]"/>
              </Container>
              <Container Size="48,20">
                <Label ID="OriginYieldCultureLabel" Color="Culture" Anchor="C,C"
Offset="0,0" Style="FontNormal14" FontStyle="Shadow"
String="+15[Icon_Culture]"/>
              </Container>
              <Container Size="48,20">
                <Label ID="OriginYieldFaithLabel" Color="Faith" Anchor="C,C"
Offset="0,0" Style="FontNormal14" FontStyle="Shadow" String="+15[Icon_Faith]"/>
              </Container>
            </Stack>
            <Stack ID="DestinationYields" Anchor="C,T" StackGrowth="Right"
StackPadding="2">
              <Container Size="48,20">
                <Label ID="DestinationYieldFoodLabel" Color="Food" Anchor="C,C"
Offset="0,0" Style="FontNormal14" FontStyle="Shadow" String="+15[Icon_Food]"/>
              </Container>
              <Container Size="48,20">
                <Label ID="DestinationYieldProductionLabel" Color="Production"
Anchor="C,C" Offset="0,0" Style="FontNormal14" FontStyle="Shadow"
String="+15[Icon_Production]"/>
              </Container>
              <Container Size="48,20">
                <Label ID="DestinationYieldGoldLabel" Color="Gold" Anchor="C,C"
Offset="0,0" Style="FontNormal14" FontStyle="Shadow" String="+15[Icon_Gold]"/>
              </Container>
              <Container Size="48,20">
                <Label ID="DestinationYieldScienceLabel" Color="Science"
Anchor="C,C" Offset="0,0" Style="FontNormal14" FontStyle="Shadow"
String="+15[Icon_Science]"/>
              </Container>
              <Container Size="48,20">
                <Label ID="DestinationYieldCultureLabel" Color="Culture"
Anchor="C,C" Offset="0,0" Style="FontNormal14" FontStyle="Shadow"
String="+15[Icon_Culture]"/>
              </Container>
```

```
              <Container Size="48,20">
                <Label ID="DestinationYieldFaithLabel" Color="Faith"
Anchor="C,C" Offset="0,0" Style="FontNormal14" FontStyle="Shadow"
String="+15[Icon_Faith]"/>
              </Container>
            </Stack>
          </Stack>
          <Button ID="CancelAutomation" Anchor="L,T" Offset="6,6" Size="21,21"
Texture="Controls_CloseButtonAltSmall" StateOffsetIncrement="0,21"
ToolTip="LOC_TRADE_CANCEL_AUTOMATION_TOOLTIP" Hidden="1"/>
      </GridButton>
    </Container>
  </Instance>
  <!-- Simple Button -->
  <Instance Name="SimpleButtonInstance">
    <Container ID="Top" Size="485,78" Offset="10,2">
      <GridButton ID="GridButton" Size="parent,parent" Color="255,255,255,150"
Style="FontNormal18">
        <GridData Texture="Controls_ListButton" SliceCorner="209,29"
SliceTextureSize="417,51" StateOffsetIncrement="0,51"/>
      </GridButton>
    </Container>
  </Instance>
    <!-- Dialog Support -->
  <Box Style="PopupDialogBox" TutorialActive="1" />
  <Instance Name="PopupButtonInstance">
    <GridButton ID="Button" Style="MainButton" Size="200,41" />
  </Instance>
  <Instance Name="PopupButtonAltInstance">
    <GridButton ID="Button" Style="ButtonRed" Size="200,41" />
  </Instance>
  <Instance Name="PopupButtonAltInstancePositive">
    <GridButton ID="Button" Style="ButtonConfirm" Size="200,41" />
  </Instance>
  <Instance Name="PopupTextInstance">
    <Label        ID="Text"       Anchor="C,T" Style="BodyTextDark18"
Align="Center" WrapWidth="400"/>
  </Instance>
  <Instance Name="RowInstance">
    <Stack ID="Row" Anchor="C,T" StackGrowth="Right" StackWrap="Bottom"
StackPadding="10" />
  </Instance>
</Context>
```

```
include( "Colors" );
-- ===========================================================================
--  Local Constants
-- ===========================================================================
local BACKDROP_DARKER_OFFSET = -85
local BACKDROP_DARKER_OPACITY = 238
local BACKDROP_BRIGHTER_OFFSET = 90
local BACKDROP_BRIGHTER_OPACITY = 250
-- set to true for faster time for UI to load, but the trader time trade route
to complete will be incorrect
-- speeds it upto 40% speedup depending upon how many long, winding routes you
have
local USE_EUCLEDIAN_DISTANCE:boolean = false
-- ===========================================================================
--  Global Constants
-- ===========================================================================
SORT_BY_ID = {
    FOOD = 1,
    PRODUCTION = 2,
    GOLD = 3,
    SCIENCE = 4,
    CULTURE = 5,
    FAITH = 6,
    TURNS_TO_COMPLETE = 7,
    ORIGIN_NAME = 8,
    DESTINATION_NAME = 9
}
SORT_ASCENDING = 1;
SORT_DESCENDING = 2;
-- Yield constants
FOOD_INDEX = GameInfo.Yields["YIELD_FOOD"].Index;
PRODUCTION_INDEX = GameInfo.Yields["YIELD_PRODUCTION"].Index;
GOLD_INDEX = GameInfo.Yields["YIELD_GOLD"].Index;
SCIENCE_INDEX = GameInfo.Yields["YIELD_SCIENCE"].Index;
CULTURE_INDEX = GameInfo.Yields["YIELD_CULTURE"].Index;
FAITH_INDEX = GameInfo.Yields["YIELD_FAITH"].Index;
START_INDEX = FOOD_INDEX;
END_INDEX = FAITH_INDEX;
-- Build lookup table for icons
ICON_LOOKUP = {}
ICON_LOOKUP[FOOD_INDEX] = "[ICON_Food]"
ICON_LOOKUP[PRODUCTION_INDEX] = "[ICON_Production]"
ICON_LOOKUP[GOLD_INDEX] = "[ICON_Gold]"
```

```
ICON_LOOKUP[SCIENCE_INDEX] = "[ICON_Science]"
ICON_LOOKUP[CULTURE_INDEX] = "[ICON_Culture]"
ICON_LOOKUP[FAITH_INDEX] = "[ICON_Faith]"
-- Build lookup table for score functions
ScoreFunctionByID = {}
ScoreFunctionByID[SORT_BY_ID.FOOD]                = function(a) return
GetYieldForOriginCity(FOOD_INDEX, a, true) end
ScoreFunctionByID[SORT_BY_ID.PRODUCTION]          = function(a) return
GetYieldForOriginCity(PRODUCTION_INDEX, a, true) end
ScoreFunctionByID[SORT_BY_ID.GOLD]                = function(a) return
GetYieldForOriginCity(GOLD_INDEX, a, true) end
ScoreFunctionByID[SORT_BY_ID.SCIENCE]             = function(a) return
GetYieldForOriginCity(SCIENCE_INDEX, a, true) end
ScoreFunctionByID[SORT_BY_ID.CULTURE]             = function(a) return
GetYieldForOriginCity(CULTURE_INDEX, a, true) end
ScoreFunctionByID[SORT_BY_ID.FAITH]               = function(a) return
GetYieldForOriginCity(FAITH_INDEX, a, true) end
ScoreFunctionByID[SORT_BY_ID.TURNS_TO_COMPLETE]   = function(a) return
GetTurnsToComplete(a, true) end
ScoreFunctionByID[SORT_BY_ID.ORIGIN_NAME]         = function(a) return
GetOriginCityName(a) end
ScoreFunctionByID[SORT_BY_ID.DESTINATION_NAME]    = function(a) return
GetDestinationCityName(a) end
-- ===========================================================================
--  Variables
-- ===========================================================================
local m_LocalPlayerRunningRoutes    :table  = {};   -- Tracks local players
active routes (turns remaining)
local m_TradersAutomatedSettings    :table  = {};   -- Tracks traders, and if
they are automated
local m_Cache                       :table  = {};   -- Cache for all route info
-- local debug_func_calls:number = 0;
-- local debug_total_calls:number = 0;
-- ===========================================================================
--  Trader Route tracker - Tracks active routes, turns remaining
-- ===========================================================================
function GetLocalPlayerRunningRoutes()
    CheckConsistencyWithMyRunningRoutes(m_LocalPlayerRunningRoutes);
    return m_LocalPlayerRunningRoutes;
end
function GetLastRouteForTrader( traderID:number )
    -- @Astog NOTE: As of Summer 2017 patch, base game added code to get this
info
```

```
    -- Commenting my modded code
    -- LoadTraderAutomatedInfo();
    -- if m_TradersAutomatedSettings[traderID] ~= nil then
    --     return m_TradersAutomatedSettings[traderID].LastRouteInfo;
    -- end
    local pTrader = Players[Game.GetLocalPlayer()]:GetUnits():FindID(traderID)
    local trade:table = pTrader:GetTrade();
    local prevOriginComponentID:table =
trade:GetLastOriginTradeCityComponentID();
    local prevDestComponentID:table =
trade:GetLastDestinationTradeCityComponentID();
    -- Make sure the entries are valid. Return nil if not
    if pTrader ~= nil and prevOriginComponentID.player ~= nil and
prevOriginComponentID.player ~= -1 and
            prevOriginComponentID.id ~= nil and prevOriginComponentID.id ~= -1
and
            prevDestComponentID.player ~= nil and prevDestComponentID.player ~=
-1 and
            prevDestComponentID.id ~= nil and prevDestComponentID.id ~= -1 then
        local routeInfo = {
            OriginCityPlayer = prevOriginComponentID.player,
            OriginCityID = prevOriginComponentID.id,
            DestinationCityPlayer = prevDestComponentID.player,
            DestinationCityID = prevDestComponentID.id
        };
        return routeInfo
    end
    return nil
end
-- Adds the route turns remaining to the table, if it does not exist already
function AddRouteWithTurnsRemaining( routeInfo:table, routesTable:table)
    -- print("Adding route: " .. GetTradeRouteString(routeInfo));
    local routeIndex = findIndex(routesTable, routeInfo, CheckRouteEquality);
    if routeIndex == -1 then
        local tradePathLength, tripsToDestination, turnsToCompleteRoute =
GetRouteInfo(routeInfo);
        -- Build entry
        local routeEntry:table = {
            OriginCityPlayer        = routeInfo.OriginCityPlayer;
            OriginCityID            = routeInfo.OriginCityID;
            DestinationCityPlayer   = routeInfo.DestinationCityPlayer;
            DestinationCityID       = routeInfo.DestinationCityID;
            TraderUnitID            = routeInfo.TraderUnitID;
```

```
            TurnsRemaining          = turnsToCompleteRoute;
        };
        -- Append entry
        table.insert(routesTable, routeEntry);
        SaveRunningRoutesInfo();
    else
        print("AddRouteWithTurnsRemaining: Route already exists in table.");
    end
end
-- Decrements routes present. Removes those that completed
function UpdateRoutesWithTurnsRemaining( routesTable:table )
    for i=1, #routesTable do
        if routesTable[i].TurnsRemaining ~= nil then
            routesTable[i].TurnsRemaining = routesTable[i].TurnsRemaining - 1;
            print("Updated route " .. GetTradeRouteString(routesTable[i]) .. "
with turns remaining " .. routesTable[i].TurnsRemaining)
        end
    end
    SaveRunningRoutesInfo();
end
-- Checks if routes running in game and the routesTable are consistent with each
other
function CheckConsistencyWithMyRunningRoutes( routesTable:table )
    -- Build currently running routes
    local routesCurrentlyRunning:table = {};
    local localPlayerCities:table = Players[Game.GetLocalPlayer()]:GetCities();
    for _, city in localPlayerCities:Members() do
        local outgoingRoutes = city:GetTrade():GetOutgoingRoutes();
        for _, routeInfo in ipairs(outgoingRoutes) do
            table.insert(routesCurrentlyRunning, routeInfo);
        end
    end
    -- Add all routes in routesCurrentlyRunning table that are not in
routesTable
    for _, route in ipairs(routesCurrentlyRunning) do
        local routeIndex = findIndex(routesTable, route, CheckRouteEquality);
        -- Is the route not present?
        if routeIndex == -1 then
            -- Add it to the list
            print(GetTradeRouteString(route) .. " was not present. Adding it to
the table.");
            AddRouteWithTurnsRemaining(route, routesTable, true);
        end
```

```
    end
    -- Remove all routes in routesTable, that are not in routesCurrentlyRunning.
    -- Manually control the indices, so that you can iterate over the table
while deleting items within it
    local i = 1;
    while i <= table.count(routesTable) do
        local routeIndex = findIndex( routesCurrentlyRunning, routesTable[i],
CheckRouteEquality );
        -- Is the route not present?
        if routeIndex == -1 then
            print("Route " .. GetTradeRouteString(routesTable[i]) .. " is no
longer running. Removing it.");
            table.remove(routesTable, i)
        else
            i = i + 1
        end
    end
    SaveRunningRoutesInfo();
end
function SaveRunningRoutesInfo()
    -- Dump active routes info
    -- print("Saving running routes info in PlayerConfig database")
    local dataDump = DataDumper(m_LocalPlayerRunningRoutes,
"localPlayerRunningRoutes");
    -- print(dataDump);
    PlayerConfigurations[Game.GetLocalPlayer()]:SetValue("BTS_LocalPlayerRunning
Rotues", dataDump);
end
function LoadRunningRoutesInfo()
    local localPlayerID = Game.GetLocalPlayer();
if(PlayerConfigurations[localPlayerID]:GetValue("BTS_LocalPlayerRunningRotues")
~= nil) then
        -- print("Retrieving previous routes PlayerConfig database")
        local dataDump =
PlayerConfigurations[localPlayerID]:GetValue("BTS_LocalPlayerRunningRotues");
        -- print(dataDump);
        loadstring(dataDump)();
        m_LocalPlayerRunningRoutes = localPlayerRunningRoutes;
    else
        print("No running route data was found, on load.")
    end
    -- Check for consistency
    CheckConsistencyWithMyRunningRoutes(m_LocalPlayerRunningRoutes);
```

```
end
-- ---------------------------------------------------------------------------
-- Game event hookups (Local to this file)
-- ---------------------------------------------------------------------------
local function TradeSupportTracker_OnUnitOperationStarted(ownerID:number,
unitID:number, operationID:number)
    if ownerID == Game.GetLocalPlayer() and operationID ==
UnitOperationTypes.MAKE_TRADE_ROUTE then
        -- Unit was just started a trade route. Find the route, and update the
tables
        local localPlayerCities:table = Players[ownerID]:GetCities();
        for _, city in localPlayerCities:Members() do
            local outgoingRoutes = city:GetTrade():GetOutgoingRoutes();
            for _, route in ipairs(outgoingRoutes) do
                if route.TraderUnitID == unitID then
                    -- Add it to the local players runnning routes
                    print("Route just started. Adding Route: " ..
GetTradeRouteString(route));
                    AddRouteWithTurnsRemaining( route,
m_LocalPlayerRunningRoutes );
                    return
                end
            end
        end
    end
end
-- Removes trader from currently running routes, when it completes
local function TradeSupportTracker_OnUnitOperationsCleared(ownerID:number,
unitID:number, operationID:number)
    if ownerID == Game.GetLocalPlayer() then
        local pPlayer:table = Players[ownerID];
        local pUnit:table = pPlayer:GetUnits():FindID(unitID);
        if pUnit ~= nil then
            local unitInfo:table = GameInfo.Units[pUnit:GetUnitType()];
            if unitInfo ~= nil and unitInfo.MakeTradeRoute then
                LoadTraderAutomatedInfo();
                -- Remove entry from local players running routes
                for _, route in ipairs(m_LocalPlayerRunningRoutes) do
                    if route.TraderUnitID == unitID then
                        if m_TradersAutomatedSettings[unitID] == nil then
                            print("Couldn't find trader automated info. Creating
one.")
                            m_TradersAutomatedSettings[unitID] = {
```

```
IsAutomated=false };
                        end
                        -- Add it to the last route info for trader
                        -- @Astog NOTE: As of Summer 2017 patch, this got added
in vanilla code, hence commenting this modded code
                        -- m_TradersAutomatedSettings[unitID].LastRouteInfo =
route;
                        -- SaveTraderAutomatedInfo();
                        print("Removing route " .. GetTradeRouteString(route) ..
" from currently running, since it completed.");
                        -- Remove route from currrently running routes
                        RemoveRouteFromTable(route, m_LocalPlayerRunningRoutes,
false);
                        SaveRunningRoutesInfo()
                        return
                    end
                end
            end
        end
    end
end
local function TradeSupportTracker_OnPlayerTurnActivated( playerID:number,
isFirstTime:boolean )
    if playerID == Game.GetLocalPlayer() and isFirstTime then
        UpdateRoutesWithTurnsRemaining(m_LocalPlayerRunningRoutes);
    end
end
-- ===========================================================================
--  Trader Route Automater - Auto renew, last route
-- ===========================================================================
function AutomateTrader(traderID:number, isAutomated:boolean,
sortSettings:table)
    LoadTraderAutomatedInfo();
    if m_TradersAutomatedSettings[traderID] == nil then
        m_TradersAutomatedSettings[traderID] = {}
    end
    m_TradersAutomatedSettings[traderID].IsAutomated = isAutomated
    if sortSettings ~= nil and table.count(sortSettings) > 0 then
        print("Automate trader " .. traderID .. " with top route.")
        m_TradersAutomatedSettings[traderID].SortSettings = sortSettings
    else
        print("Automate trader " .. traderID)
    end
```

```
    SaveTraderAutomatedInfo();
end
function CancelAutomatedTrader(traderID:number)
    print("Cancelling automation for trader " .. traderID);
    LoadTraderAutomatedInfo();
    if m_TradersAutomatedSettings[traderID] ~= nil then
        m_TradersAutomatedSettings[traderID].IsAutomated = false;
        m_TradersAutomatedSettings[traderID].SortSettings = nil;
        SaveTraderAutomatedInfo();
    else
        print("Error: Could not find automated trader info");
    end
end
function FindTopRoute(originPlayerID:number, originCityID:number,
sortSettings:table)
    local tradeManager:table = Game.GetTradeManager();
    local tradeRoutes:table = {};
    local players:table = Game.GetPlayers{ Alive=true };
    -- Build list of trade routes
    for _, player in ipairs(players) do
        local playerID = player:GetID()
        if CanPossiblyTradeWithPlayer(originPlayerID,  playerID) then
            for _, city in player:GetCities():Members() do
                local cityID = city:GetID()
                -- Can we start a trade route with this city?
                if tradeManager:CanStartRoute(originPlayerID, originCityID,
playerID, cityID) then
                    local routeInfo = {
                        OriginCityPlayer        = originPlayerID,
                        OriginCityID            = originCityID,
                        DestinationCityPlayer   = playerID,
                        DestinationCityID       = cityID
                    };
                    tradeRoutes[#tradeRoutes + 1] = routeInfo;
                end
            end
        end
    end
    -- Get the top route based on the settings saved when the route was begun.
NOTE - Will have cache misses here.
    return GetTopRouteFromSortSettings( tradeRoutes, sortSettings);
end
function RenewTradeRoutes()
```

```
    local renewedRoute:boolean = false;
    -- Load the automated settings, (so that changes from TradeOverview.lua
reach here)
    LoadTraderAutomatedInfo();
    local pPlayerUnits:table = Players[Game.GetLocalPlayer()]:GetUnits();
    for _, pUnit in pPlayerUnits:Members() do
        local unitInfo:table = GameInfo.Units[pUnit:GetUnitType()];
        local unitID:number = pUnit:GetID();
        -- Check if it is a free trader
        if unitInfo.MakeTradeRoute == true and (not
pUnit:HasPendingOperations()) then
            if m_TradersAutomatedSettings[unitID] ~= nil and
m_TradersAutomatedSettings[unitID].IsAutomated then
                local tradeManager:table = Game.GetTradeManager();
                local originCity:table = Cities.GetCityInPlot(pUnit:GetX(),
pUnit:GetY());
                local originPlayerID = originCity:GetOwner()
                local originCityID = originCity:GetID()
                local destinationPlayerID:number;
                local destinationCityID:number;
                if m_TradersAutomatedSettings[unitID].SortSettings ~= nil and
table.count(m_TradersAutomatedSettings[unitID].SortSettings) > 0 then
                    print("Picking from top sort entry");
                    -- Get destination based on the top entry
                    local topRoute = FindTopRoute(originPlayerID, originCityID,
m_TradersAutomatedSettings[unitID].SortSettings)
                    destinationPlayerID = topRoute.DestinationCityPlayer
                    destinationCityID = topRoute.DestinationCityID
                else
                    print("Picking last route");
                    local lastRouteInfo = GetLastRouteForTrader(unitID)
                    if lastRouteInfo ~= nil then
                        destinationPlayerID =
lastRouteInfo.DestinationCityPlayer
                        destinationCityID = lastRouteInfo.DestinationCityID
                    end
                end
                if tradeManager:CanStartRoute(originPlayerID, originCityID,
destinationPlayerID, destinationCityID) then
                    local destinationPlayer = Players[destinationPlayerID]
                    local destinationCity =
destinationPlayer:GetCities():FindID(destinationCityID)
                    local operationParams = {};
```

```
                    operationParams[UnitOperationTypes.PARAM_X0] =
destinationCity:GetX();
                    operationParams[UnitOperationTypes.PARAM_Y0] =
destinationCity:GetY();
                    operationParams[UnitOperationTypes.PARAM_X1] =
originCity:GetX();
                    operationParams[UnitOperationTypes.PARAM_Y1] =
originCity:GetY();
                    if (UnitManager.CanStartOperation(pUnit,
UnitOperationTypes.MAKE_TRADE_ROUTE, nil, operationParams)) then
                        print("Trader " .. unitID .. " renewed its trade route
to " .. Locale.Lookup(destinationCity:GetName()));
                        -- TODO: Send notification for renewing routes
                        UnitManager.RequestOperation(pUnit,
UnitOperationTypes.MAKE_TRADE_ROUTE, operationParams);
                        renewedRoute = true
                    else
                        print("Could not start a route");
                    end
                else
                    print("Could not renew a route. Missing route info, or the
destination is no longer a valid trade route destination.");
                end
            end
        end
    end
    -- Play sound, if a route was renewed.
    -- Done here to ensure the sound was only played once, if multiple traders
were automated
    if renewedRoute then
        UI.PlaySound("START_TRADE_ROUTE");
        SaveTraderAutomatedInfo()
    end
end
function IsTraderAutomated(traderID:number)
    LoadTraderAutomatedInfo();
    if m_TradersAutomatedSettings[traderID] ~= nil then
        return m_TradersAutomatedSettings[traderID].IsAutomated;
    end
    return false;
end
function SaveTraderAutomatedInfo()
    -- Dump active routes info
```

```
    local localPlayerID = Game.GetLocalPlayer();
    -- print("Saving Trader Automated info in PlayerConfig database")
    local dataDump = DataDumper(m_TradersAutomatedSettings,
"traderAutomatedSettings");
    -- print(dataDump);
    PlayerConfigurations[localPlayerID]:SetValue("BTS_TraderAutomatedSettings",
dataDump);
end
function LoadTraderAutomatedInfo()
    local localPlayerID = Game.GetLocalPlayer();
if(PlayerConfigurations[localPlayerID]:GetValue("BTS_TraderAutomatedSettings")
~= nil) then
        -- print("Retrieving trader automated settings from PlayerConfig
database")
        local dataDump =
PlayerConfigurations[localPlayerID]:GetValue("BTS_TraderAutomatedSettings");
        -- print(dataDump);
        loadstring(dataDump)();
        m_TradersAutomatedSettings = traderAutomatedSettings;
    else
        print("No running route data was found, on load.")
    end
end
-- ---------------------------------------------------------------------------
-- Game event hookups (Local to this file)
-- ---------------------------------------------------------------------------
local function TradeSupportAutomater_OnPlayerTurnActivated( playerID:number,
isFirstTime:boolean )
    if playerID == Game.GetLocalPlayer() and isFirstTime then
        RenewTradeRoutes();
    end
end
-- ===========================================================================
--  Cache Functions
-- ===========================================================================
function CacheRoutesInfo(tRoutes)
    if m_Cache.TurnBuilt ~= nil and m_Cache.TurnBuilt >=
Game.GetCurrentGameTurn() then
        print("OPT: Cache table already upto date")
        return false
    else
        print("Caching routes")
        -- for i, routeInfo in ipairs(tRoutes) do
```

```
        for i=1, #tRoutes do
            CacheRoute(tRoutes[i])
            CachePlayer(tRoutes[i].DestinationCityPlayer)
        end
        m_Cache.TurnBuilt = Game.GetCurrentGameTurn()
        return true
    end
end
function CacheRoute(routeInfo)
    local key:string = GetRouteKey(routeInfo);
    -- print("Key for " .. GetTradeRouteString(routeInfo) .. " is " .. key)
    m_Cache[key] = {}
    ------------------------------------------------
    -- Yields
    ------------------------------------------------
    m_Cache[key].Yields = {}
    local netOriginYield:number = 0
    local netDestinationYield:number = 0
    local originRouteYields = GetYieldForOriginCity(nil, routeInfo, false)
    local destinationRouteYields = GetYieldForDestinationCity(nil, routeInfo,
false)
    for yieldIndex = START_INDEX, END_INDEX do
        local originYield = originRouteYields[yieldIndex + 1]
        local destinationYield = destinationRouteYields[yieldIndex + 1]
        m_Cache[key].Yields[yieldIndex] = {
            Origin = originYield,
            Destination = destinationYield
        }
        netOriginYield = netOriginYield + originYield
        netDestinationYield = netDestinationYield + destinationYield
    end
    ------------------------------------------------
    -- Net Yields
    ------------------------------------------------
    m_Cache[key].NetOriginYield = netOriginYield
    m_Cache[key].NetDestinationYield = netDestinationYield
    ------------------------------------------------
    -- Trading Post
    ------------------------------------------------
    m_Cache[key].HasTradingPost = GetRouteHasTradingPost(routeInfo)
    ------------------------------------------------
    -- Advanced Info - Length, trips, turns
    ------------------------------------------------
```

```
    local tradePathLength, tripsToDestination, turnsToCompleteRoute =
GetAdvancedRouteInfo(routeInfo);
    m_Cache[key].TurnsToCompleteRoute = turnsToCompleteRoute;
    m_Cache[key].TripsToDestination = tripsToDestination;
    m_Cache[key].TradePathLength = tradePathLength;
    ------------------------------------------------
    -- Turn Built
    ------------------------------------------------
    m_Cache[key].TurnBuilt = Game.GetCurrentGameTurn()
    -- print("KEY == " .. key)
    -- dump(m_Cache[key])
end
function CachePlayer(playerID)
    -- Make entry if none exists
    if m_Cache.Players == nil then m_Cache.Players = {} end
    if m_Cache.Players[playerID] == nil then
        m_Cache.Players[playerID] = {}
        ------------------------------------------------
        -- Active Route
        ------------------------------------------------
        m_Cache.Players[playerID].HasActiveRoute = GetHasActiveRoute(playerID);
        ------------------------------------------------
        -- Visibility Index
        ------------------------------------------------
        m_Cache.Players[playerID].VisibilityIndex =
GetVisibilityIndex(playerID);
        ------------------------------------------------
        -- Icons, colors
        ------------------------------------------------
        local textureOffsetX, textureOffsetY, textureSheet, tooltip =
GetPlayerIconInfo(playerID)
        local backColor, frontColor, darkerBackColor, brighterBackColor =
GetPlayerColorInfo(playerID)
        m_Cache.Players[playerID].Icon = { textureOffsetX, textureOffsetY,
textureSheet, tooltip }
        m_Cache.Players[playerID].Colors = { backColor, frontColor,
darkerBackColor, brighterBackColor }
        ------------------------------------------------
        m_Cache.Players[playerID].TurnBuilt = Game.GetCurrentGameTurn()
    end
end
function CacheEmpty()
    -- If cache has entry TurnBuilt then the cache is built
```

```
    if m_Cache.TurnBuilt ~= nil then
        print("CACHE Emptying")
        m_Cache = {}
    end
end
function GetRouteKey(routeInfo)
    return routeInfo.OriginCityPlayer .. "_" .. routeInfo.OriginCityID .. "_" ..
                routeInfo.DestinationCityPlayer .. "_" ..
routeInfo.DestinationCityID;
end
function CacheKeyToRouteInfo(cacheKey)
    -- print("key: " .. cacheKey)
    local ids = Split(cacheKey, "_", 4) -- At max 4 entries should only exist
    local routeInfo = {
        OriginCityPlayer = tonumber(ids[1]),
        OriginCityID = tonumber(ids[2]),
        DestinationCityPlayer = tonumber(ids[3]),
        DestinationCityID = tonumber(ids[4])
    }
    -- dump(routeInfo)
    return routeInfo
end
-- ---------------------------------------------------------------------------
-- Cache lookups
-- ---------------------------------------------------------------------------
function Cached_GetYieldForOriginCity(yieldIndex:number, routeCacheKey:string)
    local cacheEntry = m_Cache[routeCacheKey]
    if cacheEntry ~= nil then
        -- print("CACHE HIT for " .. routeCacheKey)
        return cacheEntry.Yields[yieldIndex].Origin
    else
        print("CACHE MISS for " .. routeCacheKey)
        CacheRoute(CacheKeyToRouteInfo(routeCacheKey));
        return m_Cache[routeCacheKey].Yields[yieldIndex].Origin
    end
end
function Cached_GetYieldForDestinationCity(yieldIndex:number,
routeCacheKey:string)
    local cacheEntry = m_Cache[routeCacheKey]
    if cacheEntry ~= nil then
        -- print("CACHE HIT for " .. routeCacheKey)
        return cacheEntry.Yields[yieldIndex].Destination
    else
```

```
        print("CACHE MISS for " .. routeCacheKey)
        CacheRoute(CacheKeyToRouteInfo(routeCacheKey));
        return m_Cache[routeCacheKey].Yields[yieldIndex].Destination
    end
end
function Cached_GetTurnsToComplete(routeCacheKey:string)
    if m_Cache[routeCacheKey] ~= nil then
        -- print("CACHE HIT for " .. routeCacheKey)
        return m_Cache[routeCacheKey].TurnsToCompleteRoute
    else
        print("CACHE MISS for " .. routeCacheKey)
        CacheRoute(CacheKeyToRouteInfo(routeCacheKey));
        return m_Cache[routeCacheKey].TurnsToCompleteRoute
    end
end
-- ===========================================================================
--  Trade Route Sorter
-- ===========================================================================
-- This requires sort settings table passed.
function SortTradeRoutes( tradeRoutes:table, sortSettings:table)
    if table.count(sortSettings) > 0 then
        -- Score all routes based on sort settings, sort them
        local routeScores = ScoreRoutes(tradeRoutes, sortSettings)
        table.sort(routeScores, function(a, b) return ScoreComp(a, b,
sortSettings) end )
        -- Build new table based on these sorted scores
        local routes = {}
        for i, scoreInfo in ipairs(routeScores) do
            routes[i] = tradeRoutes[scoreInfo.id]
        end
        return routes
    end
    -- No sort settings, return original array
    return tradeRoutes
    -- print("Total func calls: " .. debug_func_calls)
    -- debug_total_calls = debug_total_calls + debug_func_calls
    -- print("Total calls: " .. debug_total_calls)
    -- debug_func_calls = 0;
end
function GetTopRouteFromSortSettings( tradeRoutes:table, sortSettings:table )
    if sortSettings ~= nil and table.count(sortSettings) > 0 then
        local routeScores = ScoreRoutes(tradeRoutes, sortSettings)
        local minScoreInfo = GetMinEntry(routeScores, function(a, b) return
```

```
ScoreComp(a, b, sortSettings) end )
        return tradeRoutes[minScoreInfo.id]
    end
    -- if no sort settings, return top entry
    return tradeRoutes[1];
end
-- ---------------------------------------------------------------------------
-- Score Route functions
-- ---------------------------------------------------------------------------
function ScoreRoutes( tradeRoutes:table, sortSettings:table )
    local scores = {}
    for index=1, #tradeRoutes do
        scores[index] = { id = index, score = ScoreRoute(tradeRoutes[index],
sortSettings)}
    end
    return scores
end
function ScoreRoute( routeInfo:table, sortSettings:table )
    local score = {}
    for _, sortSetting in ipairs(sortSettings) do
        local scoreFunction = ScoreFunctionByID[sortSetting.SortByID];
        local val = scoreFunction(routeInfo)
        -- Change the sign, if in descending order. EX: (-)5 < (-)2
        if sortSetting.SortOrder == SORT_DESCENDING then
            -- Handle if val is string
            if type(val) == "string" then
                val = invert_string(val)
            else
                val = val * -1
            end
        end
        score[#score + 1] = val
    end
    -- Add final score, ie net yield
    score[#score + 1] = GetNetYieldForOriginCity(routeInfo, true)
    return score
end
function ScoreComp( scoreInfo1, scoreInfo2, sortSettings )
    local score1 = scoreInfo1.score
    local score2 = scoreInfo2.score
    if #score1 ~= #score2 then
        print("ERROR = scores unequal in length")
        return false
```

```
    end
    -- Last score is the net yield, it will not have a matching sortSetting
    for i=1, #score1-1 do
        if score1[i] < score2[i] then
            return true
        elseif score1[i] > score2[i] then
            return false
        end
    end
    return score1[#score1] > score2[#score1] -- Descending order of net yield
end
-- ---------------------------------------------------------------------------
-- Sort Entries functions
-- ---------------------------------------------------------------------------
function InsertSortEntry( sortByID:number, sortOrder:number, sortSettings:table
)
    local sortEntry = {
        SortByID = sortByID,
        SortOrder = sortOrder
    };
    -- Only insert if it does not exist
    local sortEntryIndex = findIndex (sortSettings, sortEntry,
CompareSortEntries);
    if sortEntryIndex == -1 then
        -- print("Inserting " .. sortEntry.SortByID);
        table.insert(sortSettings, sortEntry);
    else
        -- If it exists, just update the sort oder
        -- print("Index: " .. sortEntryIndex);
        sortSettings[sortEntryIndex].SortOrder = sortOrder;
    end
end
function RemoveSortEntry( sortByID:number, sortSettings:table  )
    local sortEntry = {
        SortByID = sortByID,
        SortOrder = sortOrder
    };
    -- Only delete if it exists
    local sortEntryIndex:number = findIndex(sortSettings, sortEntry,
CompareSortEntries);
    if (sortEntryIndex > 0) then
        table.remove(sortSettings, sortEntryIndex);
    end
```

```
end
-- Checks for the same ID, not the same order
function CompareSortEntries( sortEntry1:table, sortEntry2:table)
    if sortEntry1.SortByID == sortEntry2.SortByID then
        return true;
    end
    return false;
end
-- ===========================================================================
--  Getter functions
-- ===========================================================================
-- Get idle Trade Units by Player ID
function GetIdleTradeUnits( playerID:number )
    local idleTradeUnits:table = {};
    -- Loop through the Players units
    local localPlayerUnits:table = Players[playerID]:GetUnits();
    for i,unit in localPlayerUnits:Members() do
        -- Find any trade units
        local unitInfo:table = GameInfo.Units[unit:GetUnitType()];
        if unitInfo.MakeTradeRoute then
            local doestradeUnitHasRoute:boolean = false;
            -- Determine if those trade units are busy by checking outgoing
routes from the players cities
            local localPlayerCities:table = Players[playerID]:GetCities();
            for _, city in localPlayerCities:Members() do
                local routes = city:GetTrade():GetOutgoingRoutes();
                for _, route in ipairs(routes) do
                    if route.TraderUnitID == unit:GetID() then
                        doestradeUnitHasRoute = true;
                    end
                end
            end
            -- If this trade unit isn't attached to an outgoing route then they
are idle
            if not doestradeUnitHasRoute then
                table.insert(idleTradeUnits, unit);
            end
        end
    end
    return idleTradeUnits;
end
-- Returns a string of the route in format
"[ORIGIN_CITY_NAME]-[DESTINATION_CITY_NAME]"
```

```
function GetTradeRouteString( routeInfo:table )
    local originCityName:string = "[NOT_FOUND]";
    local destinationCityName:string = "[NOT_FOUND]";
    local originPlayer:table = Players[routeInfo.OriginCityPlayer];
    if originPlayer ~= nil then
        local originCity:table =
originPlayer:GetCities():FindID(routeInfo.OriginCityID);
        if originCity ~= nil then
            originCityName = Locale.Lookup(originCity:GetName());
        else
            print("CITY", routeInfo.OriginCityID, "NOT FOUND")
        end
    else
        print("PLAYER", routeInfo.OriginCityPlayer, "NOT FOUND")
    end
    local destinationPlayer:table = Players[routeInfo.DestinationCityPlayer];
    if destinationPlayer ~= nil then
        local destinationCity:table =
destinationPlayer:GetCities():FindID(routeInfo.DestinationCityID);
        if destinationCity ~= nil then
            destinationCityName = Locale.Lookup(destinationCity:GetName());
        else
            print("CITY", routeInfo.DestinationCityID, "NOT FOUND")
        end
    else
        print("PLAYER", routeInfo.DestinationCityPlayer, "NOT FOUND")
    end
    return originCityName .. "-" .. destinationCityName;
end
function GetTradeRouteYieldString( routeInfo:table )
    local returnString:string = "";
    local originPlayer:table = Players[routeInfo.OriginCityPlayer];
    local originCity:table =
originPlayer:GetCities():FindID(routeInfo.OriginCityID);
    local destinationPlayer:table = Players[routeInfo.DestinationCityPlayer];
    local destinationCity:table =
destinationPlayer:GetCities():FindID(routeInfo.DestinationCityID);
    for yieldIndex = START_INDEX, END_INDEX do
        local originCityYieldValue = GetYieldForOriginCity(yieldIndex,
routeInfo, true);
        -- Skip if yield is not more than 0
        if originCityYieldValue > 0 then
            local iconString, text = FormatYieldText(yieldIndex,
```

```
originCityYieldValue);
            if (yieldIndex == FOOD_INDEX) then
                returnString = returnString .. text .. iconString .. " ";
            elseif (yieldIndex == PRODUCTION_INDEX) then
                returnString = returnString .. text .. iconString .. " ";
            elseif (yieldIndex == GOLD_INDEX) then
                returnString = returnString .. text .. iconString .. " ";
            elseif (yieldIndex == SCIENCE_INDEX) then
                returnString = returnString .. text .. iconString .. " ";
            elseif (yieldIndex == CULTURE_INDEX) then
                returnString = returnString .. text .. iconString .. " ";
            elseif (yieldIndex == FAITH_INDEX) then
                returnString = returnString .. text .. iconString;
            end
        end
    end
    return returnString;
end
-- Returns length of trade path, number of trips to destination, turns to
complete route
function GetAdvancedRouteInfo(routeInfo)
    local iSpeedCostMultiplier = GameInfo.GameSpeeds[1].CostMultiplier;
    local eSpeed = GameConfiguration.GetGameSpeedType();
    if GameInfo.GameSpeeds[eSpeed] ~= nil then
        iSpeedCostMultiplier = GameInfo.GameSpeeds[eSpeed].CostMultiplier;
    else
        print("Speed type index " .. eSpeed);
        print("Error: Could not find game speed type. Defaulting to first entry
in table");
    end
    local tradePathLength:number = 0;
    if USE_EUCLEDIAN_DISTANCE then
        local pOriginPlayer = Players[routeInfo.OriginCityPlayer]
        local pOriginCity =
pOriginPlayer:GetCities():FindID(routeInfo.OriginCityID)
        local pDestinationPlayer = Players[routeInfo.DestinationCityPlayer]
        local pDestinationCity =
pDestinationPlayer:GetCities():FindID(routeInfo.DestinationCityID)
        -- Estimate path to city using Euclidean distance
        tradePathLength = Map.GetPlotDistance(pOriginCity:GetX(),
pOriginCity:GetY(), pDestinationCity:GetX(), pDestinationCity:GetY());
    else
        -- Get exact path the trader will take
```

```
        local tradeManager = Game.GetTradeManager();
        local pathPlots =
tradeManager:GetTradeRoutePath(routeInfo.OriginCityPlayer,
routeInfo.OriginCityID, routeInfo.DestinationCityPlayer,
routeInfo.DestinationCityID);
        tradePathLength = table.count(pathPlots) - 1;
    end
    local multiplierConstant:number = 0.1;
    local tripsToDestination = 1 +
math.floor(iSpeedCostMultiplier/tradePathLength * multiplierConstant);
    local turnsToCompleteRoute = (tradePathLength * 2 * tripsToDestination);
    return tradePathLength, tripsToDestination, turnsToCompleteRoute;
end
-- ---------------------------------------------------------------------------
-- Trade Route Getters
-- ---------------------------------------------------------------------------
function GetOriginCityName( routeInfo:table )
    -- TODO - Maybe implement cache for this?
    local pPlayer = Players[routeInfo.OriginCityPlayer]
    local pCity = pPlayer:GetCities():FindID(routeInfo.OriginCityID)
    return Locale.Lookup(pCity:GetName()) -- How does lua compare localized
text?
end
function GetDestinationCityName( routeInfo:table )
    -- TODO - Maybe implement cache for this?
    local pPlayer = Players[routeInfo.DestinationCityPlayer]
    local pCity = pPlayer:GetCities():FindID(routeInfo.DestinationCityID)
    return Locale.Lookup(pCity:GetName()) -- How does lua compare localized
text?
end
-- Returns yield for the origin city
function GetYieldForOriginCity( yieldIndex:number, routeInfo:table,
checkCache:boolean )
    if checkCache then
        local key:string = GetRouteKey(routeInfo)
        return Cached_GetYieldForOriginCity(yieldIndex, key)
    else
        local tradeManager = Game.GetTradeManager();
        -- Want all the yields in a table
        if yieldIndex == nil then
            local routeYields =
tradeManager:CalculateOriginYieldsFromPotentialRoute(routeInfo.OriginCityPlayer,
routeInfo.OriginCityID, routeInfo.DestinationCityPlayer,
```

```
routeInfo.DestinationCityID);
            local pathYields =
tradeManager:CalculateOriginYieldsFromPath(routeInfo.OriginCityPlayer,
routeInfo.OriginCityID, routeInfo.DestinationCityPlayer,
routeInfo.DestinationCityID);
            local modifierYields =
tradeManager:CalculateOriginYieldsFromModifiers(routeInfo.OriginCityPlayer,
routeInfo.OriginCityID, routeInfo.DestinationCityPlayer,
routeInfo.DestinationCityID);
            -- Add the yields together and return the result
            local i;
            local yieldCount = #routeYields;
            for i=1, yieldCount, 1 do
                routeYields[i] = routeYields[i] + pathYields[i] +
modifierYields[i];
            end
            return routeYields
        else
            -- From route
            local yieldValue =
tradeManager:CalculateOriginYieldFromPotentialRoute(routeInfo.OriginCityPlayer,
routeInfo.OriginCityID, routeInfo.DestinationCityPlayer,
routeInfo.DestinationCityID, yieldIndex);
            -- From path only if yield is gold. Trading posts add only gold.
            if yieldIndex == GameInfo.Yields["YIELD_GOLD"].Index then
                yieldValue = yieldValue +
tradeManager:CalculateOriginYieldFromPath(routeInfo.OriginCityPlayer,
routeInfo.OriginCityID, routeInfo.DestinationCityPlayer,
routeInfo.DestinationCityID, yieldIndex);
            end
            -- From modifiers
            local resourceID = -1;
            yieldValue = yieldValue +
tradeManager:CalculateOriginYieldFromModifiers(routeInfo.OriginCityPlayer,
routeInfo.OriginCityID, routeInfo.DestinationCityPlayer,
routeInfo.DestinationCityID, yieldIndex, resourceID);
            return yieldValue;
        end
    end
end
-- Returns yield for the destination city
function GetYieldForDestinationCity( yieldIndex:number, routeInfo:table,
checkCache:boolean )
```

```
    if checkCache then
        local key:string = GetRouteKey(routeInfo)
        return Cached_GetYieldForDestinationCity(yieldIndex, key)
    else
        local tradeManager = Game.GetTradeManager();
        if yieldIndex == nil then
            local routeYields = tradeManager:CalculateDestinationYieldsFromPoten
tialRoute(routeInfo.OriginCityPlayer, routeInfo.OriginCityID,
routeInfo.DestinationCityPlayer, routeInfo.DestinationCityID);
            local pathYields =
tradeManager:CalculateDestinationYieldsFromPath(routeInfo.OriginCityPlayer,
routeInfo.OriginCityID, routeInfo.DestinationCityPlayer,
routeInfo.DestinationCityID);
            local modifierYields =
tradeManager:CalculateDestinationYieldsFromModifiers(routeInfo.OriginCityPlayer,
routeInfo.OriginCityID, routeInfo.DestinationCityPlayer,
routeInfo.DestinationCityID);
            -- Add the yields together and return the result
            local i;
            local yieldCount = #routeYields;
            for i=1, yieldCount, 1 do
                routeYields[i] = routeYields[i] + pathYields[i] +
modifierYields[i];
            end
            return routeYields
        else
            -- From route
            local yieldValue = tradeManager:CalculateDestinationYieldFromPotenti
alRoute(routeInfo.OriginCityPlayer, routeInfo.OriginCityID,
routeInfo.DestinationCityPlayer, routeInfo.DestinationCityID, yieldIndex);
            -- From path
            yieldValue = yieldValue +
tradeManager:CalculateDestinationYieldFromPath(routeInfo.OriginCityPlayer,
routeInfo.OriginCityID, routeInfo.DestinationCityPlayer,
routeInfo.DestinationCityID, yieldIndex);
            -- From modifiers
            local resourceID = -1;
            yieldValue = yieldValue +
tradeManager:CalculateDestinationYieldFromModifiers(routeInfo.OriginCityPlayer,
routeInfo.OriginCityID, routeInfo.DestinationCityPlayer,
routeInfo.DestinationCityID, yieldIndex, resourceID);
            return yieldValue;
        end
```

```
    end
end
function GetNetYieldForOriginCity( routeInfo, checkCache )
    if checkCache then
        local key:string = GetRouteKey(routeInfo)
        if m_Cache[key] ~= nil then
            -- print("CACHE HIT for " .. GetTradeRouteString(routeInfo))
            return m_Cache[key].NetOriginYield
        else
            print("CACHE MISS for " .. GetTradeRouteString(routeInfo))
            CacheRoute(routeInfo);
            return m_Cache[key].NetOriginYield
        end
    else
        local netYield:number = 0
        for iI = START_INDEX, END_INDEX do
            -- Dont check cache here
            netYield = netYield + GetYieldForOriginCity(iI, routeInfo)
        end
        return netYield
    end
end
function GetNetYieldForDestinationCity( routeInfo, checkCache )
    if checkCache then
        local key:string = GetRouteKey(routeInfo)
        if m_Cache[key] ~= nil then
            -- print("CACHE HIT for " .. GetTradeRouteString(routeInfo))
            return m_Cache[key].NetDestinationYield
        else
            print("CACHE MISS for " .. GetTradeRouteString(routeInfo))
            CacheRoute(routeInfo);
            return m_Cache[key].NetDestinationYield
        end
    else
        local netYield:number = 0
        for iI = START_INDEX, END_INDEX do
            -- Dont check cache here
            netYield = netYield + GetYieldForDestinationCity(iI, routeInfo)
        end
        return netYield
    end
end
function GetTurnsToComplete(routeInfo, checkCache)
```

```
    if checkCache then
        local key = GetRouteKey(routeInfo)
        if m_Cache[key] ~= nil then
            -- print("CACHE HIT for " .. GetTradeRouteString(routeInfo))
            return m_Cache[key].TurnsToCompleteRoute
        else
            print("CACHE MISS for " .. GetTradeRouteString(routeInfo))
            CacheRoute(routeInfo);
            return m_Cache[key].TurnsToCompleteRoute
        end
    else
        local tradePathLength, tripsToDestination, turnsToCompleteRoute =
GetAdvancedRouteInfo(routeInfo);
        return turnsToCompleteRoute
    end
end
function GetRouteInfo(routeInfo, checkCache)
    if checkCache then
        local key = GetRouteKey(routeInfo)
        if m_Cache[key] ~= nil then
            -- print("CACHE HIT for " .. GetTradeRouteString(routeInfo))
            return m_Cache[key].TradePathLength,
m_Cache[key].TripsToDestination, m_Cache[key].TurnsToCompleteRoute
        else
            print("CACHE MISS for " .. GetTradeRouteString(routeInfo))
            CacheRoute(routeInfo)
            return m_Cache[key].TradePathLength,
m_Cache[key].TripsToDestination, m_Cache[key].TurnsToCompleteRoute
        end
    else
        return GetAdvancedRouteInfo(routeInfo)
    end
end
function GetRouteHasTradingPost(routeInfo, checkCache)
    if checkCache then
        local key = GetRouteKey(routeInfo)
        if m_Cache[key] ~= nil then
            -- print("CACHE HIT for " .. GetTradeRouteString(routeInfo))
            return m_Cache[key].HasTradingPost
        else
            print("CACHE MISS for " .. GetTradeRouteString(routeInfo))
            CacheRoute(routeInfo)
            return m_Cache[key].HasTradingPost
```

```
        end
    else
        local destinationPlayer:table =
Players[routeInfo.DestinationCityPlayer];
        local destinationCity:table =
destinationPlayer:GetCities():FindID(routeInfo.DestinationCityID);
        return
destinationCity:GetTrade():HasActiveTradingPost(routeInfo.OriginCityPlayer)
    end
end
function GetHasActiveRoute(playerID, checkCache)
    if checkCache then
        if m_Cache.Players ~= nil and m_Cache.Players[playerID] ~= nil then
            -- print("CACHE HIT for player " .. playerID)
            return m_Cache.Players[playerID].HasActiveRoute
        else
            print("CACHE MISS for player " .. playerID)
            CachePlayer(playerID)
            return m_Cache.Players[playerID].HasActiveRoute
        end
    else
        local pPlayer:table = Players[playerID];
        local playerCities:table = pPlayer:GetCities();
        for _, city in playerCities:Members() do
            if city:GetTrade():HasTradeRouteFrom(localPlayer) then
                return true
            end
        end
        return false
    end
end
function GetVisibilityIndex(playerID, checkCache)
    if checkCache then
        if m_Cache.Players ~= nil and m_Cache.Players[playerID] ~= nil then
            -- print("CACHE HIT for player " .. playerID)
            return m_Cache.Players[playerID].VisibilityIndex
        else
            print("CACHE MISS for player " .. playerID)
            CachePlayer(playerID)
            return m_Cache.Players[playerID].VisibilityIndex
        end
    else
        return
```

```
Players[Game.GetLocalPlayer()]:GetDiplomacy():GetVisibilityOn(playerID);
    end
end
function GetPlayerIconInfo(playerID, checkCache)
    if checkCache then
        if m_Cache.Players ~= nil and m_Cache.Players[playerID] ~= nil then
            -- print("CACHE HIT for player " .. playerID)
            return unpack(m_Cache.Players[playerID].Icon)
        else
            print("CACHE MISS for player " .. playerID)
            CachePlayer(playerID)
            return unpack(m_Cache.Players[playerID].Icon)
        end
    else
        local pPlayer = Players[playerID];
        local playerConfig:table = PlayerConfigurations[playerID];
        if playerConfig ~= nil then
            local civType:string = playerConfig:GetCivilizationTypeName();
            local playerIconString:string = "ICON_" .. civType
            --[[
            if pPlayer:IsMajor() then
                -- Civilizations
                playerIconString = "ICON_" ..
playerConfig:GetCivilizationTypeName();
            else
                -- City States
                local leader:string = playerConfig:GetLeaderTypeName();
                local leaderInfo:table  = GameInfo.Leaders[leader];
                if (leader == "LEADER_MINOR_CIV_SCIENTIFIC" or
leaderInfo.InheritFrom == "LEADER_MINOR_CIV_SCIENTIFIC") then
                    playerIconString = "ICON_CITYSTATE_SCIENCE";
                elseif (leader == "LEADER_MINOR_CIV_RELIGIOUS" or
leaderInfo.InheritFrom == "LEADER_MINOR_CIV_RELIGIOUS") then
                    playerIconString = "ICON_CITYSTATE_FAITH";
                elseif (leader == "LEADER_MINOR_CIV_TRADE" or
leaderInfo.InheritFrom == "LEADER_MINOR_CIV_TRADE") then
                    playerIconString = "ICON_CITYSTATE_TRADE";
                elseif (leader == "LEADER_MINOR_CIV_CULTURAL" or
leaderInfo.InheritFrom == "LEADER_MINOR_CIV_CULTURAL") then
                    playerIconString = "ICON_CITYSTATE_CULTURE";
                elseif (leader == "LEADER_MINOR_CIV_MILITARISTIC" or
leaderInfo.InheritFrom == "LEADER_MINOR_CIV_MILITARISTIC") then
                    playerIconString = "ICON_CITYSTATE_MILITARISTIC";
```

```
                elseif (leader == "LEADER_MINOR_CIV_INDUSTRIAL" or
leaderInfo.InheritFrom == "LEADER_MINOR_CIV_INDUSTRIAL") then
                    playerIconString = "ICON_CITYSTATE_INDUSTRIAL";
                end
            end
            ]]
            local playerDescription:string =
playerConfig:GetCivilizationDescription();
            local textureOffsetX, textureOffsetY, textureSheet =
IconManager:FindIconAtlas(playerIconString, 30)
            return textureOffsetX, textureOffsetY, textureSheet,
playerDescription;
        end
    end
end
function GetPlayerColorInfo(playerID, checkCache)
    if checkCache then
        if m_Cache.Players ~= nil and m_Cache.Players[playerID] ~= nil then
            -- print("CACHE HIT for player " .. playerID)
            return unpack(m_Cache.Players[playerID].Colors)
        else
            print("CACHE MISS for player " .. playerID)
            CachePlayer(playerID)
            return unpack(m_Cache.Players[playerID].Colors)
        end
    else
        local backColor, frontColor = UI.GetPlayerColors(playerID)
        local darkerBackColor = DarkenLightenColor(backColor,
BACKDROP_DARKER_OFFSET, BACKDROP_DARKER_OPACITY);
        local brighterBackColor = DarkenLightenColor(backColor,
BACKDROP_BRIGHTER_OFFSET, BACKDROP_BRIGHTER_OPACITY);
        return backColor, frontColor, darkerBackColor, brighterBackColor
    end
end
-- ===========================================================================
--  General Helper functions
-- ===========================================================================
-- Simple check to seeif player1 and player2 can possibly have a trade route.
function CanPossiblyTradeWithPlayer(player1, player2)
    if player1 == player2 then return true; end
    local pPlayer1 = Players[player1];
    local pPlayer1Diplomacy = pPlayer1:GetDiplomacy();
    local pPlayer2 = Players[player2]
```

```
    if pPlayer2:IsAlive() and pPlayer1Diplomacy:HasMet(player2) then
        if not pPlayer1Diplomacy:IsAtWarWith(player2) then
            return true;
        end
    end
    return false;
end
function IsRoutePossible(originCityPlayerID, originCityID,
destinationCityPlayerID, destinationCityID)
    local tradeManager:table = Game.GetTradeManager();
    return tradeManager:CanStartRoute(originCityPlayerID, originCityID,
destinationCityPlayerID, destinationCityID);
end
function FormatYieldText(yieldIndex, yieldAmount)
    if yieldAmount == 0 then
        return "", ""
    end
    local iconString:string = ICON_LOOKUP[yieldIndex]
    local text:string;
    if yieldAmount > 0 then
        text = "+";
    else
        text = "-";
    end
    text = text .. Round(yieldAmount, 1);
    return iconString, text;
end
-- Finds and removes routeToDelete from routeTable
function RemoveRouteFromTable( routeToDelete:table , routeTable:table,
isGrouped:boolean )
    -- If grouping by something, go one level deeper
    if isGrouped then
        print("Routes grouped")
        local targetIndex:number = -1;
        local targetGroupIndex:number = -1;
        for i, groupedRoutes in ipairs(routeTable) do
            for j, route in ipairs(groupedRoutes) do
                if CheckRouteEquality( route, routeToDelete ) then
                    targetIndex = j;
                    targetGroupIndex = i;
                    break
                end
            end
```

```
        end
        -- Remove route
        if targetIndex ~= -1 and targetGroupIndex ~= -1 then
            print("REMOVING ROUTE")
            table.remove(routeTable[targetGroupIndex], targetIndex);
            -- If that group is empty, remove that group
            if table.count(routeTable[targetGroupIndex]) <= 0 then
                table.remove(routeTable, targetGroupIndex);
            end
        else
            print("COULD NOT FIND ROUTE")
        end
    else
        print("Routes not grouped")
        -- Find and remove route
        local targetIndex:number = findIndex(routeTable, routeToDelete,
CheckRouteEquality)
        if targetIndex ~= -1 then
            print("REMOVING ROUTE")
            table.remove(routeTable, targetIndex);
        else
            print("COULD NOT FIND ROUTE")
        end
    end
end
-- Checks if the two routes are the same (does not compare traderUnit)
function CheckRouteEquality ( tradeRoute1:table, tradeRoute2:table )
    if (    tradeRoute1.OriginCityPlayer == tradeRoute2.OriginCityPlayer and
            tradeRoute1.OriginCityID == tradeRoute2.OriginCityID and
            tradeRoute1.DestinationCityPlayer ==
tradeRoute2.DestinationCityPlayer and
            tradeRoute1.DestinationCityID == tradeRoute2.DestinationCityID )
then
        return true;
    end
    return false;
end
function IsCityState( player:table )
    local playerInfluence:table = player:GetInfluence();
    if  playerInfluence:CanReceiveInfluence() then
        return true
    end
    return false
```

```
end
-- Checks if the player is a city state, with "Send a trade route" quest
function IsCityStateWithTradeQuest( player:table )
    local questsManager:table = Game.GetQuestsManager();
    local localPlayer = Game.GetLocalPlayer()
    if (questsManager ~= nil and localPlayer ~= nil) then
        local tradeRouteQuestInfo:table =
GameInfo.Quests["QUEST_SEND_TRADE_ROUTE"];
        if (tradeRouteQuestInfo ~= nil) then
            if (questsManager:HasActiveQuestFromPlayer(localPlayer,
player:GetID(), tradeRouteQuestInfo.Index)) then
                return true
            end
        end
    end
    return false
end
-- Checks if the player is a civ, other than the local player
function IsOtherCiv( player:table )
    if player:GetID() ~= Game.GetLocalPlayer() then
        return true
    end
    return false
end
-- ===========================================================================
--  Helper Utility functions
-- ===========================================================================
-- Converts 'A' -> 'Z' || 'Z' -> 'A'
function invert_string(s:string)
    s = s:upper()
    print("org: " .. s)
    local newS:string = ""
    for i=1, s:len() do
        newS = newS .. string.char(invert_char_code(s:byte(i)))
    end
    print("inv: " .. newS)
    return newS
end
function invert_char_code(code:number)
    local delta = string.byte("Z", 1) - code
    return delta + string.byte("A", 1)
end
function table_nnill_count(T:table)
```

```
    local count = 0
    for k in pairs(T) do
        if T[k] ~= nil then
            count = count + 1
        end
    end
    return count
end
function findIndex(T, searchItem, compareFunc)
    for index, item in ipairs(T) do
        if compareFunc(item, searchItem) then
            return index;
        end
    end
    return -1;
end
function GetMinEntry(searchTable, compareFunc)
    local minIndex = 1
    for index=1, #searchTable do
        if not compareFunc(searchTable[minIndex], searchTable[index]) then
            minIndex = index;
        end
    end
    return searchTable[minIndex], minIndex
end
-- ========== START OF DataDumper.lua =================
--[[ DataDumper.lua
  Copyright (c) 2007 Olivetti-Engineering SA
  Permission is hereby granted, free of charge, to any person
  obtaining a copy of this software and associated documentation
  files (the "Software"), to deal in the Software without
  restriction, including without limitation the rights to use,
  copy, modify, merge, publish, distribute, sublicense, and/or sell
  copies of the Software, and to permit persons to whom the
  Software is furnished to do so, subject to the following
  conditions:
  The above copyright notice and this permission notice shall be
  included in all copies or substantial portions of the Software.
  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
  EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
  OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
  NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
  HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
```

```
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE SOFTWARE.
]]
function dump(...)
  print(DataDumper(...), "\n---")
end
local dumplua_closure = [[
local closures = {}
local function closure(t)
  closures[#closures+1] = t
  t[1] = assert(loadstring(t[1]))
  return t[1]
end
for _,t in pairs(closures) do
  for i = 2,#t do
    debug.setupvalue(t[1], i-1, t[i])
  end
end
]]
local lua_reserved_keywords = {
  'and', 'break', 'do', 'else', 'elseif', 'end', 'false', 'for',
  'function', 'if', 'in', 'local', 'nil', 'not', 'or', 'repeat',
  'return', 'then', 'true', 'until', 'while' }
local function keys(t)
  local res = {}
  local oktypes = { stringstring = true, numbernumber = true }
  local function cmpfct(a,b)
    if oktypes[type(a)..type(b)] then
      return a < b
    else
      return type(a) < type(b)
    end
  end
  for k in pairs(t) do
    res[#res+1] = k
  end
  table.sort(res, cmpfct)
  return res
end
local c_functions = {}
for _,lib in pairs{'_G', 'string', 'table', 'math',
    'io', 'os', 'coroutine', 'package', 'debug'} do
```

```
    local t = {}
    lib = lib .. "."
    if lib == "_G." then lib = "" end
    for k,v in pairs(t) do
      if type(v) == 'function' and not pcall(string.dump, v) then
        c_functions[v] = lib..k
      end
    end
  end
  function DataDumper(value, varname, fastmode, ident)
    local defined, dumplua = {}
    -- Local variables for speed optimization
    local string_format, type, string_dump, string_rep =
          string.format, type, string.dump, string.rep
    local tostring, pairs, table_concat =
          tostring, pairs, table.concat
    local keycache, strvalcache, out, closure_cnt = {}, {}, {}, 0
    setmetatable(strvalcache, {__index = function(t,value)
      local res = string_format('%q', value)
      t[value] = res
      return res
    end})
    local fcts = {
      string = function(value) return strvalcache[value] end,
      number = function(value) return value end,
      boolean = function(value) return tostring(value) end,
      ['nil'] = function(value) return 'nil' end,
      ['function'] = function(value)
        return string_format("loadstring(%q)", string_dump(value))
      end,
      userdata = function() error("Cannot dump userdata") end,
      thread = function() error("Cannot dump threads") end,
    }
    local function test_defined(value, path)
      if defined[value] then
        if path:match("^getmetatable.*%)$") then
          out[#out+1] = string_format("s%s, %s)\n", path:sub(2,-2),
defined[value])
        else
          out[#out+1] = path .. " = " .. defined[value] .. "\n"
        end
        return true
      end
```

```
  defined[value] = path
end
local function make_key(t, key)
  local s
  if type(key) == 'string' and key:match('^[_%a][_%w]*$') then
    s = key .. "="
  else
    s = "[" .. dumplua(key, 0) .. "]="
  end
  t[key] = s
  return s
end
for _,k in ipairs(lua_reserved_keywords) do
  keycache[k] = '["'..k..'"] = '
end
if fastmode then
  fcts.table = function (value)
    -- Table value
    local numidx = 1
    out[#out+1] = "{"
    for key,val in pairs(value) do
      if key == numidx then
        numidx = numidx + 1
      else
        out[#out+1] = keycache[key]
      end
      local str = dumplua(val)
      out[#out+1] = str..","
    end
    if string.sub(out[#out], -1) == "," then
      out[#out] = string.sub(out[#out], 1, -2);
    end
    out[#out+1] = "}"
    return ""
  end
else
  fcts.table = function (value, ident, path)
    if test_defined(value, path) then return "nil" end
    -- Table value
    local sep, str, numidx, totallen = " ", {}, 1, 0
    local meta, metastr = getmetatable(value)
    if meta then
      ident = ident + 1
```

```
          metastr = dumplua(meta, ident, "getmetatable("..path..")")
          totallen = totallen + #metastr + 16
        end
        for _,key in pairs(keys(value)) do
          local val = value[key]
          local s = ""
          local subpath = path or ""
          if key == numidx then
            subpath = subpath .. "[" .. numidx .. "]"
            numidx = numidx + 1
          else
            s = keycache[key]
            if not s:match "^%[" then subpath = subpath .. "." end
            subpath = subpath .. s:gsub("%s*=%s*$","")
          end
          s = s .. dumplua(val, ident+1, subpath)
          str[#str+1] = s
          totallen = totallen + #s + 2
        end
        if totallen > 80 then
          sep = "\n" .. string_rep("  ", ident+1)
        end
        str = "{"..sep..table_concat(str, ","..sep).." "..sep:sub(1,-3).."}"
        if meta then
          sep = sep:sub(1,-3)
          return
"setmetatable("..sep..str..","..sep..metastr..sep:sub(1,-3)..")"
        end
        return str
      end
      fcts['function'] = function (value, ident, path)
        if test_defined(value, path) then return "nil" end
        if c_functions[value] then
          return c_functions[value]
        elseif debug == nil or debug.getupvalue(value, 1) == nil then
          return string_format("loadstring(%q)", string_dump(value))
        end
        closure_cnt = closure_cnt + 1
        local res = {string.dump(value)}
        for i = 1,math.huge do
          local name, v = debug.getupvalue(value,i)
          if name == nil then break end
          res[i+1] = v
```

```
        end
        return "closure " .. dumplua(res, ident, "closures["..closure_cnt.."]")
      end
    end
    function dumplua(value, ident, path)
      return fcts[type(value)](value, ident, path)
    end
    if varname == nil then
      varname = ""
    elseif varname:match("^[%a_][%w_]*$") then
      varname = varname .. " = "
    end
    if fastmode then
      setmetatable(keycache, {__index = make_key })
      out[1] = varname
      table.insert(out,dumplua(value, 0))
      return table.concat(out)
    else
      setmetatable(keycache, {__index = make_key })
      local items = {}
      for i=1,10 do items[i] = '' end
      items[3] = dumplua(value, ident or 0, "t")
      if closure_cnt > 0 then
        items[1], items[6] = dumplua_closure:match("(.*\n)\n(.*)")
        out[#out+1] = ""
      end
      if #out > 0 then
        items[2], items[4] = "local t = ", "\n"
        items[5] = table.concat(out)
        items[7] = varname .. "t"
      else
        items[2] = varname
      end
      return table.concat(items)
    end
  end
-- ========== END OF DataDumper.lua ================
-- ============================================================================
--  Event handlers
-- ============================================================================
function TradeSupportTracker_Initialize()
    print("Initializing BTS Trade Support Tracker");
    -- Load Previous Routes
```

```
    LoadRunningRoutesInfo();
    Events.UnitOperationStarted.Add( TradeSupportTracker_OnUnitOperationStarted
);
    Events.UnitOperationsCleared.Add(
TradeSupportTracker_OnUnitOperationsCleared );
    Events.PlayerTurnActivated.Add( TradeSupportTracker_OnPlayerTurnActivated );
end
function TradeSupportAutomater_Initialize()
    print("Initializing BTS Trade Support Automater");
    -- Load previous automated settings
    LoadTraderAutomatedInfo();
    Events.PlayerTurnActivated.Add( TradeSupportAutomater_OnPlayerTurnActivated
);
end
```

```
-- ===========================================================================
--
--  Slideout panel that allows the player to move their trade units to other
city centers
--
-- ===========================================================================
include("InstanceManager");
include("SupportFunctions");
include("AnimSidePanelSupport");
include("Colors")
-- ===========================================================================
--  CONSTANTS
-- ===========================================================================
local RELOAD_CACHE_ID:string = "TradeOriginChooser"; -- Must be unique (usually
the same as the file name)
-- ===========================================================================
--  MEMBERS
-- ===========================================================================
local m_AnimSupport:table; --AnimSidePanelSupport
local m_cityIM:table = InstanceManager:new("CityInstance", "CityButton",
Controls.CityStack);
local m_originCity = nil;
local m_newOriginCity = nil;
-- ===========================================================================
function Refresh()
    -- Find the selected trade unit
    local selectedUnit:table = UI.GetHeadSelectedUnit();
    if selectedUnit == nil then
        Close();
        return;
    end
    -- Find the current city
    m_originCity = Cities.GetCityInPlot(selectedUnit:GetX(),
selectedUnit:GetY());
    if m_originCity == nil then
        Close();
        return;
    end
    RefreshHeader();
    -- Reset Instance Manager
    m_cityIM:ResetInstances();
    local cityIDs:table = {}
    -- Add all other cities to city stack
```

```
    local localPlayer = Players[Game.GetLocalPlayer()];
    local playerCities:table = localPlayer:GetCities();
    for _, city in playerCities:Members() do
        if city ~= m_originCity and CanTeleportToCity(city) then
            table.insert(cityIDs, city:GetID())
        end
    end
    -- Sort cities alphabetically
    local function comp(a, b)
        local playerCities = Players[Game.GetLocalPlayer()]:GetCities()
        local city1 = playerCities:FindID(a)
        local city2 = playerCities:FindID(b)
        return Locale.Lookup(city1:GetName()):upper() <
Locale.Lookup(city2:GetName()):upper()
    end
    table.sort(cityIDs, comp)
    for _, cityID in ipairs(cityIDs) do
        AddCity(cityID)
    end
    -- Calculate Control Size
    Controls.CityStack:CalculateSize();
    Controls.CityScrollPanel:CalculateSize();
end
-- ===========================================================================
function RefreshHeader()
    if m_newOriginCity then
        Controls.BannerBase:SetHide(false);
        Controls.ChangeOriginCityButton:SetHide(false);
        Controls.StatusMessage:SetHide(true);
        Controls.CityName:SetText(Locale.ToUpper(m_newOriginCity:GetName()));
        -- Update City Banner
        local backColor:number, frontColor:number  = UI.GetPlayerColors(
m_newOriginCity:GetOwner() );
        local darkerBackColor:number = DarkenLightenColor(backColor,(-85),238);
        local brighterBackColor:number = DarkenLightenColor(backColor,90,255);
        Controls.BannerBase:SetColor( backColor );
        Controls.BannerDarker:SetColor( darkerBackColor );
        Controls.BannerLighter:SetColor( brighterBackColor );
        Controls.CityName:SetColor( frontColor );
        -- Update Icon
        local originPlayerConfig:table =
PlayerConfigurations[m_newOriginCity:GetOwner()];
        local originPlayerIconString:string = "ICON_" ..
```

```
originPlayerConfig:GetCivilizationTypeName();
        local textureOffsetX, textureOffsetY, textureSheet =
IconManager:FindIconAtlas(originPlayerIconString, 22);
        local secondaryColor, primaryColor = UI.GetPlayerColors(
m_newOriginCity:GetOwner() );
        local brighterIconColor:number =
DarkenLightenColor(primaryColor,90,255);
        Controls.OriginCivIcon:SetTexture(textureOffsetX, textureOffsetY,
textureSheet);
        Controls.OriginCivIcon:LocalizeAndSetToolTip(
originPlayerConfig:GetCivilizationDescription() );
        Controls.OriginCivIcon:SetColor( primaryColor );
    else
        Controls.BannerBase:SetHide(true);
        Controls.ChangeOriginCityButton:SetHide(true);
        Controls.StatusMessage:SetHide(false);
        Controls.StatusMessage:SetText(Locale.Lookup("LOC_ORIGIN_CHOOSER_HEADER_
BACKGROUND_TEXT"));
    end
end
-- ===========================================================================
function AddCity(cityID:number)
    local city = Players[Game.GetLocalPlayer()]:GetCities():FindID(cityID)
    print("Adding city " .. Locale.Lookup(city:GetName()))
    local cityInstance:table = m_cityIM:GetInstance();
    cityInstance.CityButton:SetHide(false);
    cityInstance.CityButton:SetText(Locale.ToUpper(city:GetName()));
    if m_newOriginCity ~= nil and m_newOriginCity:GetID() == cityID then
        cityInstance.CityButton:SetTextureOffsetVal(0, 32*1)
    else
        cityInstance.CityButton:SetTextureOffsetVal(0, 32*0)
    end
    cityInstance.CityButton:RegisterCallback(Mouse.eLClick,
        function()
            m_newOriginCity = city;
            Refresh();
        end);
end
-- ===========================================================================
function OnChangeOriginCityButton()
    if ( m_newOriginCity ~= nil and m_originCity ~= nil ) then
        if ( m_newOriginCity:GetID() ~= m_originCity:GetID() ) then
            TeleportToCity(m_newOriginCity);
```

```
        else
            -- print (" cant teleport to the same city")
        end
    else
        print("cities are nil")
    end
end
-- ===========================================================================
function CanTeleportToCity(city:table)
    local tParameters = {};
    tParameters[UnitOperationTypes.PARAM_X] = city:GetX();
    tParameters[UnitOperationTypes.PARAM_Y] = city:GetY();
    -- local eOperation =
UI.GetInterfaceModeParameter(UnitOperationTypes.PARAM_OPERATION_TYPE);
    local eOperation = UnitOperationTypes.TELEPORT_TO_CITY
    local pSelectedUnit = UI.GetHeadSelectedUnit();
    if (UnitManager.CanStartOperation( pSelectedUnit, eOperation, nil,
tParameters)) then
        return true;
    end
    return false;
end
-- ===========================================================================
function TeleportToCity(city:table)
    local tParameters = {};
    tParameters[UnitOperationTypes.PARAM_X] = city:GetX();
    tParameters[UnitOperationTypes.PARAM_Y] = city:GetY();
    -- local eOperation =
UI.GetInterfaceModeParameter(UnitOperationTypes.PARAM_OPERATION_TYPE);
    local eOperation = UnitOperationTypes.TELEPORT_TO_CITY
    local pSelectedUnit = UI.GetHeadSelectedUnit();
    if (UnitManager.CanStartOperation( pSelectedUnit, eOperation, nil,
tParameters)) then
        UnitManager.RequestOperation( pSelectedUnit, eOperation, tParameters);
        UI.SetInterfaceMode(InterfaceModeTypes.SELECTION);
        UI.PlaySound("Unit_Relocate");
        OnClose();
    end
end
-- ===========================================================================
function OnChangeOriginCityFromOverview( city:table )
    if city ~= nil then
        -- print ("Window opened from Trade Overview with city " ..
```

```
Locale.Lookup(city:GetName()))
        local selectedUnit:table = UI.GetHeadSelectedUnit();
        m_originCity = Cities.GetCityInPlot(selectedUnit:GetX(),
selectedUnit:GetY());
        m_newOriginCity = city
        -- print ("Transfer from " .. Locale.Lookup(m_originCity:GetName()) .. "
to " .. Locale.Lookup(m_newOriginCity:GetName()))
        -- Is the screen already open?
        if (m_AnimSupport:IsVisible()) then
            Refresh();
        else
            print("open sesame...")
            OnOpen();
        end
    end
end
-- ===========================================================================
function OnInterfaceModeChanged( oldMode:number, newMode:number )
    if (oldMode == InterfaceModeTypes.TELEPORT_TO_CITY) then
        -- Only close if already open
        if m_AnimSupport:IsVisible() then
            Close();
        end
    end
    if (newMode == InterfaceModeTypes.MAKE_TRADE_ROUTE) then
        -- Only close if already open
        if m_AnimSupport:IsVisible() then
            Close();
        end
        UILens.SetActive("TradeRoute");
    end
    if (newMode == InterfaceModeTypes.TELEPORT_TO_CITY) then
        -- Only open if selected unit is a trade unit
        local pSelectedUnit:table = UI.GetHeadSelectedUnit();
        local pSelectedUnitInfo:table =
GameInfo.Units[pSelectedUnit:GetUnitType()];
        if pSelectedUnitInfo.MakeTradeRoute then
            Open();
        end
    end
end
-- ===========================================================================
function OnCitySelectionChanged(owner, ID, i, j, k, bSelected, bEditable)
```

```
    -- Close if we select a city
    if m_AnimSupport:IsVisible() and owner == Game.GetLocalPlayer() and owner ~=
-1 then
        OnClose();
    end
end
-- ===========================================================================
function OnUnitSelectionChanged( playerID:number, unitID:number, hexI:number,
hexJ:number, hexK:number, bSelected:boolean, bEditable:boolean)
    -- Check if the unit selected is a trader. Don't do anything if it is
    local selectedUnit:table = Players[playerID]:GetUnits():FindID(unitID)
    if selectedUnit ~= nil then
        local selectedUnitInfo:table =
GameInfo.Units[selectedUnit:GetUnitType()];
        if selectedUnitInfo ~= nil and selectedUnitInfo.MakeTradeRoute == true
then
            local activityType:number =
UnitManager.GetActivityType(selectedUnit);
            if activityType == ActivityTypes.ACTIVITY_AWAKE and
selectedUnit:GetMovesRemaining() > 0 then
                return -- early return here so OnClose() is not called
            end
        end
    end
    -- Close if screen shown
    if m_AnimSupport:IsVisible() and playerID == Game.GetLocalPlayer() and
playerID ~= -1 then
        OnClose()
    end
end
-----------------------------------------------------------------------------
----------------
function OnLocalPlayerTurnEnd()
    if GameConfiguration.IsHotseat() then
        OnClose();
    end
end
-- ===========================================================================
function Open()
    LuaEvents.TradeOriginChooser_SetTradeUnitStatus("LOC_HUD_UNIT_PANEL_CHOOSING
_ORIGIN_CITY");
    m_AnimSupport:Show();
    Refresh();
```

```
end
-- ===========================================================================
function Close()
    LuaEvents.TradeOriginChooser_SetTradeUnitStatus("");
    m_AnimSupport:Hide();
    m_originCity = nil
    m_newOriginCity = nil
    -- Switch to default Lens
    -- UILens.SetActive("Default"); -- Done when lens is turned off
end
-- ===========================================================================
function OnOpen()
    LuaEvents.TradeRouteChooser_Close()
    Open();
end
-- ===========================================================================
function OnClose()
    if UI.GetInterfaceMode() == InterfaceModeTypes.TELEPORT_TO_CITY then
        UI.SetInterfaceMode(InterfaceModeTypes.SELECTION);
    elseif m_AnimSupport:IsVisible() then
        Close()
    end
end
-- ===========================================================================
--  Input
--  UI Event Handler
-- ===========================================================================
function KeyDownHandler( key:number )
    return false;
end
function KeyUpHandler( key:number )
    if key == Keys.VK_RETURN then
        OnChangeOriginCityButton()
        -- Dont let it fall through
        return true;
    end
    if key == Keys.VK_ESCAPE then
        OnClose();
        -- Dont let it fall through
        return true;
    end
    return false;
end
```

```
function OnInputHandler( pInputStruct:table )
    local uiMsg = pInputStruct:GetMessageType();
    local catchEvent = false
    if uiMsg == KeyEvents.KeyDown then
        catchEvent = KeyDownHandler( pInputStruct:GetKey() )
    end
    if uiMsg == KeyEvents.KeyUp then
        catchEvent = KeyUpHandler( pInputStruct:GetKey() )
    end
    if not catchEvent then
        return m_AnimSupport.OnInputHandler(pInputStruct)
    end
    return catchEvent
end
-- ===========================================================================
--  HOT-RELOADING EVENTS
-- ===========================================================================
function OnInit(isReload:boolean)
    if isReload then
        LuaEvents.GameDebug_GetValues(RELOAD_CACHE_ID);
    end
end
function OnShutdown()
    LuaEvents.GameDebug_AddValue(RELOAD_CACHE_ID, "isVisible",
m_AnimSupport:IsVisible());
end
function OnGameDebugReturn(context:string, contextTable:table)
    if context == RELOAD_CACHE_ID and contextTable["isVisible"] ~= nil and
contextTable["isVisible"] then
        OnOpen();
    end
end
-- ===========================================================================
--  INIT
-- ===========================================================================
function Initialize()
    print("Initializing BTS Trade Origin Chooser");
    -- Hot-reload events
    ContextPtr:SetInitHandler(OnInit);
    ContextPtr:SetShutdown(OnShutdown);
    LuaEvents.GameDebug_Return.Add(OnGameDebugReturn);
    LuaEvents.TradeOverview_ChangeOriginCityFromOverview.Add(
OnChangeOriginCityFromOverview );
```

```
    -- Game Engine Events
    Events.InterfaceModeChanged.Add( OnInterfaceModeChanged );
    Events.CitySelectionChanged.Add( OnCitySelectionChanged );
    Events.UnitSelectionChanged.Add( OnUnitSelectionChanged );
    Events.LocalPlayerTurnEnd.Add( OnLocalPlayerTurnEnd );
    -- Animation controller
    m_AnimSupport = CreateScreenAnimation(Controls.SlideAnim);
    -- Animation controller events
    Events.SystemUpdateUI.Add(m_AnimSupport.OnUpdateUI);
    ContextPtr:SetInputHandler( OnInputHandler, true );
    -- Control Events
    Controls.CloseButton:RegisterCallback(Mouse.eLClick, OnClose);
    Controls.CloseButton:RegisterCallback( Mouse.eMouseEnter, function()
UI.PlaySound("Main_Menu_Mouse_Over"); end);
    Controls.ChangeOriginCityButton:RegisterCallback(Mouse.eLClick,
OnChangeOriginCityButton);
    Controls.ChangeOriginCityButton:RegisterCallback( Mouse.eMouseEnter,
function() UI.PlaySound("Main_Menu_Mouse_Over"); end);
end
Initialize();
```

```
<?xml version="1.0" encoding="utf-8"?>
<Context xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSc
hemaLocation="..\..\..\..\..\CivTech\Libs\ForgeUI\ForgeUI_Assets\Controls.xsd">
  <SlideAnim Style="ChooserAnim">
    <Stack ID="RouteOriginChooser" StackGrowth="Bottom">
      <!-- Header Container -->
      <Container ID="RouteOriginChooserHeader" Size="parent,150">
        <Label ID="RouteOriginChooserHeaderText" Size="parent,25" Offset="-6,10"
Anchor="C,T" Style="FontFlair14" SmallCaps="20" SmallCapsType="EveryWord"
Color="0,59,77,255" String="LOC_TRADE_ORIGIN_CHOOSER_HEADER_LABEL_TEXT"/>
        <Grid Size="parent+5,parent-25" Anchor="C,B" Offset="1,0"
Texture="DestinationChooser_CurrentSlot" SliceStart="0,0" SliceCorner="30,30"
SliceSize="250,42" SliceTextureSize="309,172" >
        </Grid>
        <Button ID="CloseButton" Anchor="R,T" Offset="-11,-1"
Style="CloseButtonSmall"/>
        <!-- City Banner -->
        <Grid ID="BannerBase" Anchor="C,T" Offset="0,40" Size="270,33"
Texture="CityPanel_BannerBase" SliceCorner="20,10" SliceSize="160,1"
SliceTextureSize="199,33" Color="150,170,100,255">
          <Image ID="OriginCivIcon" Anchor="L,C" Size="22,22" Offset="10,-4"
Icon="ICON_CIVILIZATION_UNKNOWN" IconSize="30"/>
          <Grid ID="BannerDarker" Anchor="L,T" Offset="4,2"
Size="parent-8,parent-10" Texture="CityPanel_BannerDarker" SliceCorner="95,10"
SliceSize="1,1" SliceTextureSize="191,23" Color="0,0,0,100" />
          <Grid ID="BannerLighter" Anchor="L,T" Offset="4,2"
Size="parent-8,parent-10" Texture="CityPanel_BannerLighter" SliceCorner="95,10"
SliceSize="1,1" SliceTextureSize="191,23" Color="255,255,255,255" />
          <Grid Anchor="L,T" Offset="6,2" Size="parent-10,parent-8"
Texture="CityPanel_BannerNone" SliceCorner="70,10" SliceSize="1,1"
SliceTextureSize="179,20" Color="255,0,0,255" />
          <Label ID="CityName" Anchor="L,C" Offset="40,-2" Style="FontFlair16"
FontStyle="Stroke" EffectColor="0,0,0,25" String="$CityName$" SmallCaps="20"
SmallCapsType="EveryWord" TruncateWidth="220" />
        </Grid>
        <!-- Status Message -->
        <Container Size="parent, parent" Offset="0,13">
          <Label ID="StatusMessage" Size="300,50" Anchor="C,C"
Style="FontNormal16" Color="0,0,0,150"/>
        </Container>
        <!-- Confirm Button -->
        <GridButton ID="ChangeOriginCityButton" Size="parent-35,26"
Offset="1,24" Anchor="C,B" String="LOC_ORIGIN_CHOOSER_CHANGE_DESTINATION_BUTTON"
```

```
Style="FontFlair14" FontStyle="Shadows" EffectColor="0,0,0,255" TextOffset="0,2"
SmallCaps="20" SmallCapsType="EveryWord">
          <GridData Texture="Controls_ConfirmSmall" StateOffsetIncrement="0,26"
SliceCorner="26,13" SliceSize="2,2" SliceTextureSize="51,26"/>
        </GridButton>
      </Container>
      <Container ID="RouteOriginChooserBody" Anchor="L,T" Offset="20,5"
Size="parent-40,parent-160">
        <ScrollPanel ID="CityScrollPanel" Anchor="L,T" Offset="0,0"
Size="parent+10,parent" Vertical="1" AutoScrollBar="0" AutoSizeScrollBar="1">
          <ScrollBar    Anchor="L,T" Size="11,parent" AnchorSide="O,I"
Offset="4,0" Style="ScrollVerticalBar" />
          <Stack ID="CityStack" Anchor="L,T" Offset="5,0" StackGrowth="Down"/>
        </ScrollPanel>
      </Container>
    </Stack>
  </SlideAnim>
  <!-- Instances -->
  <Instance Name="CityInstance">
    <GridButton ID="CityButton" Size="parent,30" Style="FontFlair18"
SmallCaps="22" SmallCapsType="EveryWord" TextAnchor="L,C" TextOffset="30,0">
      <GridData Style="ButtonLightWeightGrid"/>
    </GridButton>
  </Instance>
</Context>
```

```
-- ===========================================================================
--  Settings
-- ===========================================================================
local showSortOrdersPermanently = false
local RoutePanelBaseOffsetX = 8;
local RoutePanelScrollPanelExtraOffset = 9;
-- ===========================================================================
--  INCLUDES
-- ===========================================================================
include("InstanceManager");
include("SupportFunctions");
include("TradeSupport");
-- ===========================================================================
--  VARIABLES
-- ===========================================================================
local m_RouteChoiceIM           : table =
InstanceManager:new("RouteChoiceInstance", "Top", Controls.RouteChoiceStack);
local m_originCity              : table = nil;  -- City where the trade route
will begin
local m_destinationCity         : table = nil;  -- City where the trade route
will end, nil if none selected
local m_TradeRouteLens:number = UILens.CreateLensLayerHash("TradeRoutes");
-- These can be set by other contexts to have a route selected automatically
after the chooser opens
local m_postOpenSelectPlayerID:number = -1;
local m_postOpenSelectCityID:number = -1;
local m_AvailableTradeRoutes:table = {}; -- Filtered and unfiltered lists of
possible routes
local m_TradeRoutes:table = {} -- Routes showm, this is the filtered and sorted
local m_TurnBuiltRouteTable:number = -1;
local m_LastTrader:number = -1;
local m_RebuildAvailableRoutes:boolean = true;
-- Stores filter list and tracks the currently selected list
local m_filterList:table = {};
local m_filterCount:number = 0;
local m_filterSelected:number = 1;
local m_shiftDown:boolean = false;
-- Stores the sort settings.
local m_SortBySettings:table = {};
local m_SortSettingsChanged:boolean = true;
local m_FilterSettingsChanged:boolean = true;
local m_SkipNextOpen:boolean = false;
-- Default is ascending in turns to complete trade route
```

```
m_SortBySettings[1] = {
    SortByID = SORT_BY_ID.TURNS_TO_COMPLETE,
    SortOrder = SORT_ASCENDING
};
local opt_print = false
-- ===========================================================================
--  Refresh functions
-- ===========================================================================
function Refresh()
    local selectedUnit:table = UI.GetHeadSelectedUnit();
    if selectedUnit == nil then
        Close();
        return;
    end
    m_originCity = Cities.GetCityInPlot(selectedUnit:GetX(),
selectedUnit:GetY());
    if m_originCity == nil then
        Close();
        return;
    end
    -- Rebuild if turn has advanced or unit has changed
    if m_LastTrader ~= selectedUnit:GetID() or m_TurnBuiltRouteTable <
Game.GetCurrentGameTurn() then
        m_LastTrader = selectedUnit:GetID()
        -- Rebuild and re-sort
        m_RebuildAvailableRoutes = true
    else
        m_RebuildAvailableRoutes = false
    end
    -- Handle post open (ie TradeOverview) calls
    if m_postOpenSelectPlayerID ~= -1 and m_postOpenSelectCityID ~= -1 then
        print("Selecting", m_postOpenSelectCityID)
        local pPlayer = Players[m_postOpenSelectPlayerID]
        m_destinationCity = pPlayer:GetCities():FindID(m_postOpenSelectCityID)
        RealizeLookAtDestinationCity();
        -- Reset values
        m_postOpenSelectPlayerID = -1;
        m_postOpenSelectCityID = -1;
    end
    RefreshHeader();
    RefreshTopPanel();
    RefreshSortBar();
    RefreshChooserPanel();
```

```
end
function RefreshHeader()
    if m_originCity then
        Controls.Header_OriginText:SetText(Locale.Lookup("LOC_ROUTECHOOSER_TO_DE
STINATION", Locale.ToUpper(m_originCity:GetName())));
    end
end
function RefreshTopPanel()
    if m_destinationCity and m_originCity then
        local tradeRoute = {
            OriginCityPlayer        = m_originCity:GetOwner(),
            OriginCityID            = m_originCity:GetID(),
            DestinationCityPlayer   = m_destinationCity:GetOwner(),
            DestinationCityID       = m_destinationCity:GetID()
        };
        -- Update City Banner
        Controls.CityName:SetText(Locale.ToUpper(m_destinationCity:GetName()));
        local backColor, frontColor, darkerBackColor, brighterBackColor =
GetPlayerColorInfo(m_destinationCity:GetOwner(), true);
        Controls.BannerBase:SetColor(backColor);
        Controls.BannerDarker:SetColor(darkerBackColor);
        Controls.BannerLighter:SetColor(brighterBackColor);
        Controls.CityName:SetColor(frontColor);
        -- Update Trading Post Icon
        if GetRouteHasTradingPost(tradeRoute, true) then
            Controls.TradingPostIcon:SetHide(false);
        else
            Controls.TradingPostIcon:SetHide(true);
        end
        -- Update City-State Quest Icon
        Controls.CityStateQuestIcon:SetHide(true);
        local questsManager : table = Game.GetQuestsManager();
        local questTooltip  : string = Locale.Lookup("LOC_CITY_STATES_QUESTS");
        if (questsManager ~= nil and Game.GetLocalPlayer() ~= nil) then
            local tradeRouteQuestInfo:table =
GameInfo.Quests["QUEST_SEND_TRADE_ROUTE"];
            if (tradeRouteQuestInfo ~= nil) then
                if
(questsManager:HasActiveQuestFromPlayer(Game.GetLocalPlayer(),
m_destinationCity:GetOwner(), tradeRouteQuestInfo.Index)) then
                    questTooltip = questTooltip .. "[NEWLINE]" ..
tradeRouteQuestInfo.IconString ..
questsManager:GetActiveQuestName(Game.GetLocalPlayer(),
```

```
m_destinationCity:GetOwner(), tradeRouteQuestInfo.Index);
                    Controls.CityStateQuestIcon:SetHide(false);
                    Controls.CityStateQuestIcon:SetToolTipString(questTooltip);
                end
            end
        end
        -- Update turns to complete route
        local tradePathLength, tripsToDestination, turnsToCompleteRoute =
GetRouteInfo(tradeRoute, true);
        Controls.TurnsToComplete:SetColor(frontColor);
        Controls.TurnsToComplete:SetText(turnsToCompleteRoute);
        -- Update Resources
        Controls.OriginResourceList:DestroyAllChildren();
        Controls.DestinationResourceList:DestroyAllChildren();
        local originYieldInstance:table = {};
        local originReceivedResources:boolean = false;
        local destinationYieldInstance:table = {};
        local destinationReceivedResources:boolean = false;
        ContextPtr:BuildInstanceForControl( "RouteYieldInstance",
originYieldInstance, Controls.OriginResourceList );
        ContextPtr:BuildInstanceForControl( "RouteYieldInstance",
destinationYieldInstance, Controls.DestinationResourceList );
        for yieldIndex = START_INDEX, END_INDEX do
            local originCityYieldValue = GetYieldForOriginCity(yieldIndex,
tradeRoute, true);
            local destinationCityYieldValue =
GetYieldForDestinationCity(yieldIndex, tradeRoute, true);
            SetRouteInstanceYields(originYieldInstance, yieldIndex,
originCityYieldValue);
            SetRouteInstanceYields(destinationYieldInstance, yieldIndex,
destinationCityYieldValue);
            if not originReceivedResources and originCityYieldValue > 0 then
                originReceivedResources = true
            end
            if not destinationReceivedResources and destinationCityYieldValue >
0 then
                destinationReceivedResources = true
            end
        end
        Controls.OriginResourceHeader:SetText(Locale.Lookup("LOC_ROUTECHOOSER_RE
CEIVES_RESOURCE", Locale.Lookup(m_originCity:GetName())));
        Controls.DestinationResourceHeader:SetText(Locale.Lookup("LOC_ROUTECHOOS
ER_RECEIVES_RESOURCE", Locale.Lookup(m_destinationCity:GetName())));
```

```
        local originTooltipText = ""
        local destinationTooltipText:string = "";
        -- Handle Religion pressure icons
        local destinationMajorityReligion =
m_destinationCity:GetReligion():GetMajorityReligion();
        if (destinationMajorityReligion > 0) then
            local pressureValue, sourceText =
GetReligiousPressureForCity(destinationMajorityReligion, m_destinationCity,
true);
            if (pressureValue ~= 0) then
                if (originTooltipText ~= "") then
                    originTooltipText = originTooltipText .. "[NEWLINE]";
                end
                originTooltipText = originTooltipText .. sourceText;
                AddReligiousPressureResourceEntry(GameInfo.Religions[destination
MajorityReligion], pressureValue, true, sourceText, originYieldInstance);
                originReceivedResources = true;
            end
        end
        Controls.OriginResources:SetToolTipString(originTooltipText);
        local originMajorityReligion =
m_originCity:GetReligion():GetMajorityReligion();
        if (originMajorityReligion > 0) then
            local pressureValue, sourceText =
GetReligiousPressureForCity(originMajorityReligion, m_originCity, false);
            if (pressureValue ~= 0) then
                if (destinationTooltipText ~= "") then
                    destinationTooltipText = destinationTooltipText ..
"[NEWLINE]";
                end
                destinationTooltipText = destinationTooltipText .. sourceText;
AddReligiousPressureResourceEntry(GameInfo.Religions[originMajorityReligion],
pressureValue, false, sourceText, destinationYieldInstance);
                destinationReceivedResources = true;
            end
        end
        Controls.DestinationResources:SetToolTipString(destinationTooltipText);
        if originReceivedResources then
            Controls.OriginReceivesNoBenefitsLabel:SetHide(true);
        else
            Controls.OriginReceivesNoBenefitsLabel:SetHide(false);
        end
        if destinationReceivedResources then
```

```
            Controls.DestinationReceivesNoBenefitsLabel:SetHide(true);
        else
            Controls.DestinationReceivesNoBenefitsLabel:SetHide(false);
        end
        -- Cleanup
        Controls.OriginResourceList:CalculateSize();
        Controls.OriginResourceList:ReprocessAnchoring();
        Controls.DestinationResourceList:CalculateSize();
        Controls.DestinationResourceList:ReprocessAnchoring();
        Controls.TopGrid:DoAutoSize();
        -- Show Panel
        Controls.CurrentSelectionContainer:SetHide(false);
        Controls.CurrentSelectionContainer:DoAutoSize();
        -- Hide Status Message
        Controls.StatusMessage:SetHide(true);
    else
        -- Hide Panel
        Controls.CurrentSelectionContainer:SetHide(true);
        -- Show Status Message
        Controls.StatusMessage:SetHide(false);
        Controls.StatusMessage:DoAutoSize();
    end
end
function RefreshChooserPanel()
    local tradeManager:table = Game.GetTradeManager();
    -- Do we rebuild available routes?
    if m_RebuildAvailableRoutes then
        -- Reset Available routes
        m_AvailableTradeRoutes = {};
        -- Gather available routes
        local originCityPlayerID = m_originCity:GetOwner()
        local originCityID = m_originCity:GetID()
        local players:table = Game.GetPlayers{ Alive=true };
        for _, player in ipairs(players) do
            local destinationCityPlayerID = player:GetID()
            for _, city in player:GetCities():Members() do
                local destinationCityID = city:GetID()
                -- Can we start a trade route with this city?
                if tradeManager:CanStartRoute(originCityPlayerID, originCityID,
destinationCityPlayerID, destinationCityID) then
                    local tradeRoute = {
                        OriginCityPlayer        = originCityPlayerID,
                        OriginCityID            = originCityID,
```

```
                        DestinationCityPlayer   = destinationCityPlayerID,
                        DestinationCityID       = destinationCityID
                    };
                    table.insert(m_AvailableTradeRoutes, tradeRoute);
                end
            end
        end
        -- Need to re-filter and re-sort
        m_SortSettingsChanged = true
        m_FilterSettingsChanged = true
        -- Cache routes info.
        CacheEmpty()
        CacheRoutesInfo(m_AvailableTradeRoutes)
        m_TurnBuiltRouteTable = Game.GetCurrentGameTurn()
        m_RebuildAvailableRoutes = false -- done building routes
    else
        if opt_print then
            print("OPT: Not rebuilding routes")
        end
    end
    -- Update Filters
    RefreshFilters();
    -- Update Destination Choice Stack
    RefreshStack();
    -- Send Trade Route Paths to Engine
    UILens.ClearLayerHexes( m_TradeRouteLens );
    local DEFAULT_TINT = RGBAValuesToABGRHex(1, 1, 1, 1);
    local FADED_TINT = RGBAValuesToABGRHex(0.3, 0.3, 0.3, 1);
    -- If a city is selected, fade the other routes
    local kUnselectedColor = DEFAULT_TINT;
    if (m_destinationCity ~= nil) then kUnselectedColor = FADED_TINT; end
    -- Show all paths that aren't selected
    local pathPlots:table = {};
    for _, routeInfo in ipairs(m_TradeRoutes) do
        local destinationPlayer:table =
Players[routeInfo.DestinationCityPlayer];
        local destinationCity:table =
destinationPlayer:GetCities():FindID(routeInfo.DestinationCityID);
        pathPlots = tradeManager:GetTradeRoutePath(m_originCity:GetOwner(),
m_originCity:GetID(), destinationCity:GetOwner(), destinationCity:GetID() );
        local kVariations:table = {};
        local lastElement:number = table.count(pathPlots);
        table.insert(kVariations, {"TradeRoute_Destination",
```

```
pathPlots[lastElement]} );
        if (destinationCity ~= m_destinationCity) then
            UILens.SetLayerHexesPath( m_TradeRouteLens, Game.GetLocalPlayer(),
pathPlots, kVariations, kUnselectedColor );
        end
    end
    -- Show the selected path last if it exists so it's on top
    if m_destinationCity ~= nil then
        pathPlots = tradeManager:GetTradeRoutePath(m_originCity:GetOwner(),
m_originCity:GetID(), m_destinationCity:GetOwner(), m_destinationCity:GetID() );
        local kVariations:table = {};
        local lastElement : number = table.count(pathPlots);
        table.insert(kVariations, {"TradeRoute_Destination",
pathPlots[lastElement]} );
        UILens.SetLayerHexesPath( m_TradeRouteLens, Game.GetLocalPlayer(),
pathPlots, kVariations, DEFAULT_TINT );
    end
end
-- ===========================================================================
--  Routes stack Function
-- ===========================================================================
function RefreshStack()
    -- Reset destinations
    m_RouteChoiceIM:ResetInstances();
    local tradeManager:table = Game.GetTradeManager();
    -- Filter Destinations by active Filter
    if m_FilterSettingsChanged then
        m_TradeRoutes = FilterTradeRoutes(m_AvailableTradeRoutes);
        m_FilterSettingsChanged = false -- done filtering
        -- Filter changed, need to re-sort
        m_SortSettingsChanged = true
    else
        if opt_print then
            print("OPT: Not refiltering.")
        end
    end
    if m_SortSettingsChanged then
        m_TradeRoutes = SortTradeRoutes(m_TradeRoutes, m_SortBySettings);
        m_SortSettingsChanged = false -- done sorting
    else
        if opt_print then
            print("OPT: Not resorting.")
        end
```

```
    end
    -- for i, tradeRoute in ipairs(tradeRoutes) do
    for i=1, #m_TradeRoutes do
        AddRouteToDestinationStack(m_TradeRoutes[i]);
    end
    Controls.RouteChoiceStack:CalculateSize();
    Controls.RouteChoiceScrollPanel:CalculateSize();
    -- Adjust offset based on scroll bar
    if Controls.RouteChoiceScrollPanel:GetScrollBar():IsHidden() then
        Controls.RouteContainer:SetOffsetX(RoutePanelBaseOffsetX);
    else
        Controls.RouteContainer:SetOffsetX(RoutePanelBaseOffsetX +
RoutePanelScrollPanelExtraOffset);
    end
    -- Show No Available Trade Routes message if nothing to select
    if #m_TradeRoutes > 0 then
        Controls.StatusMessage:SetText(Locale.Lookup("LOC_ROUTECHOOSER_SELECT_DE
STINATION"));
    else
        Controls.StatusMessage:SetText(Locale.Lookup("LOC_ROUTECHOOSER_NO_TRADE_
ROUTES"));
    end
end
function AddRouteToDestinationStack(routeInfo:table)
    local cityEntry:table = m_RouteChoiceIM:GetInstance();
    local destinationPlayer:table = Players[routeInfo.DestinationCityPlayer];
    local destinationCity:table =
destinationPlayer:GetCities():FindID(routeInfo.DestinationCityID);
    local originPlayer:table = Players[routeInfo.OriginCityPlayer];
    local originCity:table =
originPlayer:GetCities():FindID(routeInfo.OriginCityID);
    -- Update Selector Brace
    if m_destinationCity ~= nil and destinationCity:GetName() ==
m_destinationCity:GetName() then
        cityEntry.SelectorBrace:SetHide(false);
        cityEntry.Button:SetTextureOffsetVal(0, 76*1)
    else
        cityEntry.SelectorBrace:SetHide(true);
        cityEntry.Button:SetTextureOffsetVal(0, 76*0)
    end
    -- Setup city banner
    cityEntry.CityName:SetText(Locale.ToUpper(destinationCity:GetName()));
    local backColor, frontColor, darkerBackColor, brighterBackColor =
```

```
GetPlayerColorInfo(routeInfo.DestinationCityPlayer, true);
    cityEntry.BannerBase:SetColor(backColor);
    cityEntry.BannerDarker:SetColor(darkerBackColor);
    cityEntry.BannerLighter:SetColor(brighterBackColor);
    cityEntry.CityName:SetColor(frontColor);
    -- Update Trading Post Icon
    if GetRouteHasTradingPost(routeInfo, true) then
        cityEntry.TradingPostIcon:SetHide(false);
    else
        cityEntry.TradingPostIcon:SetHide(true);
    end
    -- Update City-State Quest Icon
    cityEntry.CityStateQuestIcon:SetHide(true);
    local questsManager : table = Game.GetQuestsManager();
    local questTooltip  : string = Locale.Lookup("LOC_CITY_STATES_QUESTS");
    if (questsManager ~= nil and Game.GetLocalPlayer() ~= nil) then
        local tradeRouteQuestInfo:table =
GameInfo.Quests["QUEST_SEND_TRADE_ROUTE"];
        if (tradeRouteQuestInfo ~= nil) then
            if
(questsManager:HasActiveQuestFromPlayer(routeInfo.OriginCityPlayer,
routeInfo.DestinationCityPlayer, tradeRouteQuestInfo.Index)) then
                questTooltip = questTooltip .. "[NEWLINE]" ..
tradeRouteQuestInfo.IconString ..
questsManager:GetActiveQuestName(Game.GetLocalPlayer(),
routeInfo.DestinationCityPlayer, tradeRouteQuestInfo.Index);
                cityEntry.CityStateQuestIcon:SetHide(false);
                cityEntry.CityStateQuestIcon:SetToolTipString(questTooltip);
            end
        end
    end
    local tradePathLength, tripsToDestination, turnsToCompleteRoute =
GetRouteInfo(routeInfo, true);
    tooltipString = (   Locale.Lookup("LOC_TRADE_TURNS_REMAINING_HELP_TOOLTIP")
.. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_TOOLTIP_BREAKER") .. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_ROUTE_LENGTH_TOOLTIP", tradePathLength)
.. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_TRIPS_COUNT_TOOLTIP",
tripsToDestination) .. "[NEWLINE]" ..
Locale.Lookup("LOC_TRADE_TURNS_REMAINING_TURN_COMPLETION_ALT_TOOLTIP",
turnsToCompleteRoute, (Game.GetCurrentGameTurn() + turnsToCompleteRoute)) );
    cityEntry.TurnsToComplete:SetText(turnsToCompleteRoute);
```

```
    cityEntry.TurnsToComplete:SetToolTipString( tooltipString );
    cityEntry.TurnsToComplete:SetColor( frontColor );
    -- Setup resources
    local tooltipText = "";
    cityEntry.ResourceList:DestroyAllChildren();
    local originYieldInstance:table = {};
    local destinationYieldInstance:table = {};
    ContextPtr:BuildInstanceForControl( "RouteYieldInstance",
originYieldInstance, cityEntry.ResourceList );
    ContextPtr:BuildInstanceForControl( "RouteYieldInstance",
destinationYieldInstance, cityEntry.ResourceList );
    for yieldIndex = START_INDEX, END_INDEX do
        -- Don't used a cache call here, since we need more info for the tooltip
        local originYieldValue, sourceText = GetYieldForCity(yieldIndex,
destinationCity, true);
        -- Normal cached call here
        local destinationYieldValue = GetYieldForDestinationCity(yieldIndex,
routeInfo, true);
        if originYieldValue > 0 then
            if (tooltipText ~= "" and originYieldValue > 0) then
                tooltipText = tooltipText .. "[NEWLINE]";
            end
            tooltipText = tooltipText .. sourceText;
        end
        SetRouteInstanceYields(originYieldInstance, yieldIndex,
originYieldValue)
        SetRouteInstanceYields(destinationYieldInstance, yieldIndex,
destinationYieldValue)
    end
    local destinationMajorityReligion =
destinationCity:GetReligion():GetMajorityReligion();
    if (destinationMajorityReligion > 0) then
        local pressureValue, sourceText =
GetReligiousPressureForCity(destinationMajorityReligion, destinationCity, true);
        if (pressureValue ~= 0) then
            if (tooltipText ~= "") then
                tooltipText = tooltipText .. "[NEWLINE]";
            end
            tooltipText = tooltipText .. sourceText;
            AddReligiousPressureResourceEntry(GameInfo.Religions[destinationMajo
rityReligion], pressureValue, true, sourceText, originYieldInstance);
        end
    end
```

```
    local originMajorityReligion =
originCity:GetReligion():GetMajorityReligion();
    if (originMajorityReligion > 0) then
        local pressureValue, sourceText =
GetReligiousPressureForCity(originMajorityReligion, destinationCity, false);
        if (pressureValue ~= 0) then
            if (tooltipText ~= "") then
                tooltipText = tooltipText .. "[NEWLINE]";
            end
            tooltipText = tooltipText .. sourceText;
AddReligiousPressureResourceEntry(GameInfo.Religions[originMajorityReligion],
pressureValue, false, sourceText, destinationYieldInstance);
        end
    end
    -- Cleanup
    cityEntry.ResourceList:CalculateSize();
    cityEntry.ResourceList:ReprocessAnchoring();
    cityEntry.Button:SetToolTipString(tooltipText);
    -- Setup callback
    cityEntry.Button:SetVoids(routeInfo.DestinationCityPlayer,
routeInfo.DestinationCityID);
    cityEntry.Button:RegisterCallback( Mouse.eLClick, OnTradeRouteSelected );
end
-- ---------------------------------------------------------------------------
-- Route button helpers
-- ---------------------------------------------------------------------------
-- ===========================================================================
function SetRouteInstanceYields(yieldsInstance, yieldIndex, yieldValue)
    local iconString, text = FormatYieldText(yieldIndex, yieldValue);
    if (yieldIndex == FOOD_INDEX) then
        yieldsInstance.YieldFoodLabel:SetText(text .. iconString);
    elseif (yieldIndex == PRODUCTION_INDEX) then
        yieldsInstance.YieldProductionLabel:SetText(text .. iconString);
    elseif (yieldIndex == GOLD_INDEX) then
        yieldsInstance.YieldGoldLabel:SetText(text .. iconString);
    elseif (yieldIndex == SCIENCE_INDEX) then
        yieldsInstance.YieldScienceLabel:SetText(text .. iconString);
    elseif (yieldIndex == CULTURE_INDEX) then
        yieldsInstance.YieldCultureLabel:SetText(text .. iconString);
    elseif (yieldIndex == FAITH_INDEX) then
        yieldsInstance.YieldFaithLabel:SetText(text .. iconString);
    end
end
```

```
-- ===========================================================================
function GetReligiousPressureForCity(religionIndex:number,
destinationCity:table, forOriginCity:boolean)
    local pressureValue = 0;
    local pressureIconString = "";
    local cityName = "";
    local tradeManager = Game.GetTradeManager();
    if m_originCity == nil or destinationCity == nil then
        return 0, "";
    end
    if (forOriginCity) then
        pressureValue = tradeManager:CalculateOriginReligiousPressureFromPotenti
alRoute(m_originCity:GetOwner(), m_originCity:GetID(),
destinationCity:GetOwner(), destinationCity:GetID(), religionIndex);
        pressureIconString = "[ICON_PressureLeft]";
        cityName = destinationCity:GetName();
    else
        pressureValue = tradeManager:CalculateDestinationReligiousPressureFromPo
tentialRoute(m_originCity:GetOwner(), m_originCity:GetID(),
destinationCity:GetOwner(), destinationCity:GetID(), religionIndex);
        pressureIconString = "[ICON_PressureRight]";
        cityName = m_originCity:GetName();
    end
    local sourceText =
Locale.Lookup("LOC_ROUTECHOOSER_RELIGIOUS_PRESSURE_SOURCE_MAJORITY_RELIGION",
pressureValue, pressureIconString, Game.GetReligion():GetName(religionIndex),
cityName);
    return pressureValue, sourceText;
end
-- ===========================================================================
function AddReligiousPressureResourceEntry(religionInfo:table,
pressureValue:number, forOriginCity:boolean, sourceText:string,
instanceControl:table)
    -- local entryInstance:table = {};
    -- ContextPtr:BuildInstanceForControl( "ReligionPressureEntryInstance",
entryInstance, stackControl );
    instanceControl.RouteReligionContainer:SetHide(false);
    local religionColor = UI.GetColorValue(religionInfo.Color);
    local religionName = Game.GetReligion():GetName(religionInfo.Index);
    instanceControl.ReligionIcon:SetIcon("ICON_" .. religionInfo.ReligionType);
    instanceControl.ReligionIcon:SetColor(religionColor);
    instanceControl.ReligionIconBacking:SetColor(religionColor);
    instanceControl.ReligionIconBacking:SetToolTipString(religionName);
```

```
    local icon:string, text:string = FormatReligiousPressureText(religionInfo,
pressureValue, forOriginCity);
    instanceControl.ResourceEntryText:SetText(text);
    -- instanceControl.RouteReligionContainer:CalculateSize();
    -- instanceControl.RouteReligionContainer:ReprocessAnchoring();
end
-- ===========================================================================
function FormatReligiousPressureText(religionInfo, pressureValue,
forOriginCity:boolean)
    local text:string = "";
    local iconString = "";
    if (religionInfo ~= nil) then
        if (forOriginCity) then
            iconString = "[ICON_PressureLeft]";
        else
            iconString = "[ICON_PressureRight]";
        end
    end
    if (pressureValue >= 0) then
        text = text .. "+";
    end
    text = text .. pressureValue;
    return iconString, text;
end
-- ===========================================================================
--  Filter, Filter Pulldown functions
-- ===========================================================================
function FilterTradeRoutes ( tradeRoutes:table )
    -- print("Current filter: " .. m_filterList[m_filterSelected].FilterText);
    if m_filterSelected == 1 then
        return tradeRoutes;
    end
    local filtertedRoutes:table = {};
    for index, tradeRoute in ipairs(tradeRoutes) do
        local pPlayer = Players[tradeRoute.DestinationCityPlayer];
        if m_filterList[m_filterSelected].FilterFunction and
m_filterList[m_filterSelected].FilterFunction(pPlayer) then
            table.insert(filtertedRoutes, tradeRoute);
        end
    end
    return filtertedRoutes;
end
-- ---------------------------------------------------------------------------
```

```
-- Filter pulldown functions
-- ---------------------------------------------------------------------------
function RefreshFilters()
    -- Clear current filters
    Controls.DestinationFilterPulldown:ClearEntries();
    m_filterList = {};
    m_filterCount = 0;
    -- Add "All" Filter
    AddFilter(Locale.Lookup("LOC_ROUTECHOOSER_FILTER_ALL"), function(a) return
true; end);
    -- Add "International Routes" Filter
    AddFilter(Locale.Lookup("LOC_TRADE_FILTER_INTERNATIONAL_ROUTES_TEXT") ,
IsOtherCiv);
    -- Add "City States with Trade Quest" Filter
    AddFilter(Locale.Lookup("LOC_TRADE_FILTER_CS_WITH_QUEST_TOOLTIP"),
IsCityStateWithTradeQuest);
    -- Add Local Player Filter
    local localPlayerConfig:table = PlayerConfigurations[Game.GetLocalPlayer()];
    local localPlayerName = Locale.Lookup(GameInfo.Civilizations[localPlayerConf
ig:GetCivilizationTypeID()].Name);
    AddFilter(localPlayerName, function(a) return a:GetID() ==
Game.GetLocalPlayer(); end);
    -- Add Filters by Civ
    local players:table = Game.GetPlayers();
    for index, pPlayer in ipairs(players) do
        if pPlayer and pPlayer:IsAlive() and pPlayer:IsMajor() then
            -- Has the local player met the civ?
            if pPlayer:GetDiplomacy():HasMet(Game.GetLocalPlayer()) then
                local playerConfig:table =
PlayerConfigurations[pPlayer:GetID()];
                local name = Locale.Lookup(GameInfo.Civilizations[playerConfig:G
etCivilizationTypeID()].Name);
                AddFilter(name, function(a) return a:GetID() == pPlayer:GetID()
end);
            end
        end
    end
    -- Add "City States" Filter
    AddFilter(Locale.Lookup("LOC_HUD_REPORTS_CITY_STATE"), IsCityState);
    -- Add filters to pulldown
    for index, filter in ipairs(m_filterList) do
        AddFilterEntry(index);
    end
```

```
    -- Select first filter
    Controls.FilterButton:SetText(m_filterList[m_filterSelected].FilterText);
    -- Calculate Internals
    Controls.DestinationFilterPulldown:CalculateInternals();
    UpdateFilterArrow();
end
function AddFilter( filterName:string, filterFunction )
    -- Make sure we don't add duplicate filters
    for index, filter in ipairs(m_filterList) do
        if filter.FilterText == filterName then
            return;
        end
    end
    m_filterCount = m_filterCount + 1;
    m_filterList[m_filterCount] = {FilterText=filterName,
FilterFunction=filterFunction};
end
function AddFilterEntry( filterIndex:number )
    local filterEntry:table = {};
    Controls.DestinationFilterPulldown:BuildEntry( "FilterEntry", filterEntry );
    filterEntry.Button:SetText(m_filterList[filterIndex].FilterText);
    filterEntry.Button:SetVoids(i, filterIndex);
end
function UpdateFilterArrow()
    if Controls.DestinationFilterPulldown:IsOpen() then
        Controls.PulldownOpenedArrow:SetHide(true);
        Controls.PulldownClosedArrow:SetHide(false);
    else
        Controls.PulldownOpenedArrow:SetHide(false);
        Controls.PulldownClosedArrow:SetHide(true);
    end
end
function OnFilterSelected( index:number, filterIndex:number )
    m_filterSelected = filterIndex;
    Controls.FilterButton:SetText(m_filterList[m_filterSelected].FilterText);
    m_FilterSettingsChanged = true
    Refresh();
end
-- ===========================================================================
--  Sort bar functions
-- ===========================================================================
-- Hides all the ascending/descending arrows
function ResetSortBar()
```

```
    Controls.FoodDescArrow:SetHide(true);
    Controls.ProductionDescArrow:SetHide(true);
    Controls.GoldDescArrow:SetHide(true);
    Controls.ScienceDescArrow:SetHide(true);
    Controls.CultureDescArrow:SetHide(true);
    Controls.FaithDescArrow:SetHide(true);
    Controls.TurnsToCompleteDescArrow:SetHide(true);
    Controls.FoodAscArrow:SetHide(true);
    Controls.ProductionAscArrow:SetHide(true);
    Controls.GoldAscArrow:SetHide(true);
    Controls.ScienceAscArrow:SetHide(true);
    Controls.CultureAscArrow:SetHide(true);
    Controls.FaithAscArrow:SetHide(true);
    Controls.TurnsToCompleteAscArrow:SetHide(true);
end
function RefreshSortBar()
    RefreshSortButtons( m_SortBySettings );
    if showSortOrdersPermanently or m_shiftDown then
        -- Hide the order texts
        HideSortOrderLabels();
        -- Show them based on current settings
        ShowSortOrderLabels();
    end
end
function ShowSortOrderLabels()
    -- Refresh and show sort orders
    RefreshSortOrderLabels( m_SortBySettings );
end
function HideSortOrderLabels()
    Controls.FoodSortOrder:SetHide(true);
    Controls.ProductionSortOrder:SetHide(true);
    Controls.GoldSortOrder:SetHide(true);
    Controls.ScienceSortOrder:SetHide(true);
    Controls.CultureSortOrder:SetHide(true);
    Controls.FaithSortOrder:SetHide(true);
    Controls.TurnsToCompleteSortOrder:SetHide(true);
end
-- Shows and hides arrows based on the passed sort order
function SetSortArrow( ascArrow:table, descArrow:table, sortOrder:number )
    if sortOrder == SORT_ASCENDING then
        descArrow:SetHide(true);
        ascArrow:SetHide(false);
    else
```

```
        descArrow:SetHide(false);
        ascArrow:SetHide(true);
    end
end
function RefreshSortButtons( sortSettings:table )
    -- Hide all arrows
    ResetSortBar();
    -- Set disabled color
    Controls.FoodSortButton:SetColorByName("ButtonDisabledCS");
    Controls.ProductionSortButton:SetColorByName("ButtonDisabledCS");
    Controls.GoldSortButton:SetColorByName("ButtonDisabledCS");
    Controls.ScienceSortButton:SetColorByName("ButtonDisabledCS");
    Controls.CultureSortButton:SetColorByName("ButtonDisabledCS");
    Controls.FaithSortButton:SetColorByName("ButtonDisabledCS");
    Controls.TurnsToCompleteSortButton:SetColorByName("ButtonDisabledCS");
    -- Go through settings and display arrows
    for index, sortEntry in ipairs(sortSettings) do
        if sortEntry.SortByID == SORT_BY_ID.FOOD then
            SetSortArrow(Controls.FoodAscArrow, Controls.FoodDescArrow,
sortEntry.SortOrder)
            Controls.FoodSortButton:SetColorByName("ButtonCS");
        elseif sortEntry.SortByID == SORT_BY_ID.PRODUCTION then
            SetSortArrow(Controls.ProductionAscArrow,
Controls.ProductionDescArrow, sortEntry.SortOrder)
            Controls.ProductionSortButton:SetColorByName("ButtonCS");
        elseif sortEntry.SortByID == SORT_BY_ID.GOLD then
            SetSortArrow(Controls.GoldAscArrow, Controls.GoldDescArrow,
sortEntry.SortOrder)
            Controls.GoldSortButton:SetColorByName("ButtonCS");
        elseif sortEntry.SortByID == SORT_BY_ID.SCIENCE then
            SetSortArrow(Controls.ScienceAscArrow, Controls.ScienceDescArrow,
sortEntry.SortOrder)
            Controls.ScienceSortButton:SetColorByName("ButtonCS");
        elseif sortEntry.SortByID == SORT_BY_ID.CULTURE then
            SetSortArrow(Controls.CultureAscArrow, Controls.CultureDescArrow,
sortEntry.SortOrder)
            Controls.CultureSortButton:SetColorByName("ButtonCS");
        elseif sortEntry.SortByID == SORT_BY_ID.FAITH then
            SetSortArrow(Controls.FaithAscArrow, Controls.FaithDescArrow,
sortEntry.SortOrder)
            Controls.FaithSortButton:SetColorByName("ButtonCS");
        elseif sortEntry.SortByID == SORT_BY_ID.TURNS_TO_COMPLETE then
            SetSortArrow(Controls.TurnsToCompleteAscArrow,
```

```
Controls.TurnsToCompleteDescArrow, sortEntry.SortOrder)
            Controls.TurnsToCompleteSortButton:SetColorByName("ButtonCS");
        end
    end
end
function RefreshSortOrderLabels( sortSettings:table )
    for index, sortEntry in ipairs(sortSettings) do
        if sortEntry.SortByID == SORT_BY_ID.FOOD then
            Controls.FoodSortOrder:SetHide(false);
            Controls.FoodSortOrder:SetText(index);
            Controls.FoodSortOrder:SetColorByName("ResFoodLabelCS");
        elseif sortEntry.SortByID == SORT_BY_ID.PRODUCTION then
            Controls.ProductionSortOrder:SetHide(false);
            Controls.ProductionSortOrder:SetText(index);
            Controls.ProductionSortOrder:SetColorByName("ResProductionLabelCS");
        elseif sortEntry.SortByID == SORT_BY_ID.GOLD then
            Controls.GoldSortOrder:SetHide(false);
            Controls.GoldSortOrder:SetText(index);
            Controls.GoldSortOrder:SetColorByName("ResGoldLabelCS");
        elseif sortEntry.SortByID == SORT_BY_ID.SCIENCE then
            Controls.ScienceSortOrder:SetHide(false);
            Controls.ScienceSortOrder:SetText(index);
            Controls.ScienceSortOrder:SetColorByName("ResScienceLabelCS");
        elseif sortEntry.SortByID == SORT_BY_ID.CULTURE then
            Controls.CultureSortOrder:SetHide(false);
            Controls.CultureSortOrder:SetText(index);
            Controls.CultureSortOrder:SetColorByName("ResCultureLabelCS");
        elseif sortEntry.SortByID == SORT_BY_ID.FAITH then
            Controls.FaithSortOrder:SetHide(false);
            Controls.FaithSortOrder:SetText(index);
            Controls.FaithSortOrder:SetColorByName("ResFaithLabelCS");
        elseif sortEntry.SortByID == SORT_BY_ID.TURNS_TO_COMPLETE then
            Controls.TurnsToCompleteSortOrder:SetHide(false);
            Controls.TurnsToCompleteSortOrder:SetText(index);
        end
    end
end
-- ===========================================================================
--  General Helper functions
-- ===========================================================================
-- ---------------------------------------------------------------------------
-- Trade route helper functions
-- ---------------------------------------------------------------------------
```

```
function TradeRouteSelected( cityOwner:number, cityID:number )
    local player:table = Players[cityOwner];
    if player then
        local pCity:table = player:GetCities():FindID(cityID);
        if pCity ~= nil then
            m_destinationCity = pCity;
        else
            error("Unable to find city '".. tostring(cityID).."' for creating a
trade route.");
        end
    end
    Refresh();
end
function GetYieldForCity(yieldIndex:number, city:table, originCity:boolean)
    local tradeManager = Game.GetTradeManager();
    local yieldInfo = GameInfo.Yields[yieldIndex];
    local totalValue = 0;
    local partialValue = 0;
    local sourceText = "";
    -- From route
    if (originCity) then
        partialValue =
tradeManager:CalculateOriginYieldFromPotentialRoute(m_originCity:GetOwner(),
m_originCity:GetID(), city:GetOwner(), city:GetID(), yieldIndex);
    else
        partialValue = tradeManager:CalculateDestinationYieldFromPotentialRoute(
m_originCity:GetOwner(), m_originCity:GetID(), city:GetOwner(), city:GetID(),
yieldIndex);
    end
    totalValue = totalValue + partialValue;
    if (partialValue > 0 and yieldInfo ~= nil) then
        if (sourceText ~= "") then
            sourceText = sourceText .. "[NEWLINE]";
        end
        sourceText = sourceText ..
Locale.Lookup("LOC_ROUTECHOOSER_YIELD_SOURCE_DISTRICTS", partialValue,
yieldInfo.IconString, yieldInfo.Name, city:GetName());
    end
    -- From path
    if (originCity) then
        partialValue =
tradeManager:CalculateOriginYieldFromPath(m_originCity:GetOwner(),
m_originCity:GetID(), city:GetOwner(), city:GetID(), yieldIndex);
```

```
    else
        partialValue =
tradeManager:CalculateDestinationYieldFromPath(m_originCity:GetOwner(),
m_originCity:GetID(), city:GetOwner(), city:GetID(), yieldIndex);
    end
    totalValue = totalValue + partialValue;
    if (partialValue > 0 and yieldInfo ~= nil) then
        if (sourceText ~= "") then
            sourceText = sourceText .. "[NEWLINE]";
        end
        sourceText = sourceText ..
Locale.Lookup("LOC_ROUTECHOOSER_YIELD_SOURCE_TRADING_POSTS", partialValue,
yieldInfo.IconString, yieldInfo.Name);
    end
    -- From modifiers
    local resourceID = -1;
    if (originCity) then
        partialValue =
tradeManager:CalculateOriginYieldFromModifiers(m_originCity:GetOwner(),
m_originCity:GetID(), city:GetOwner(), city:GetID(), yieldIndex, resourceID);
    else
        partialValue =
tradeManager:CalculateDestinationYieldFromModifiers(m_originCity:GetOwner(),
m_originCity:GetID(), city:GetOwner(), city:GetID(), yieldIndex, resourceID);
    end
    totalValue = totalValue + partialValue;
    if (partialValue > 0 and yieldInfo ~= nil) then
        if (sourceText ~= "") then
            sourceText = sourceText .. "[NEWLINE]";
        end
        sourceText = sourceText ..
Locale.Lookup("LOC_ROUTECHOOSER_YIELD_SOURCE_BONUSES", partialValue,
yieldInfo.IconString, yieldInfo.Name);
    end
    return totalValue, sourceText;
end
-- ===========================================================================
--  Look at the plot of the destination city.
--  Not always done when selected, as sometimes the TradeOverview will be
--  open and it's going to perform it's own lookat.
-- ===========================================================================
function RealizeLookAtDestinationCity()
    if m_destinationCity == nil then
```

```
        UI.DataError("TradeRouteChooser cannot look at a NIL destination.");
        return;
    end
    local locX      :number = m_destinationCity:GetX();
    local locY      :number = m_destinationCity:GetY();
    local screenXOff:number = 0.6;
    -- Change offset if the TradeOveriew (exists and) is open as well.
    local pContextControl:table =
ContextPtr:LookUpControl("/InGame/TradeOverview");
    if pContextControl == nil then
        UI.DataError("Cannot determine if partial screen
\"/InGame/TradeOverview\" is visible because it wasn't found at that path.");
    elseif not pContextControl:IsHidden() then
        screenXOff = 0.42;
    end
    UI.LookAtPlotScreenPosition( locX, locY, screenXOff, 0.5 ); -- Look at 60%
over from left side of screen
end
-- ===========================================================================
--  UI Button Callback
-- ===========================================================================
function OnTradeRouteSelected( cityOwner:number, cityID:number )
    TradeRouteSelected( cityOwner, cityID );
    RealizeLookAtDestinationCity();
    LuaEvents.TradeRouteChooser_RouteConsidered();
end
function OnRepeatRouteCheckbox()
    if not Controls.RepeatRouteCheckbox:IsChecked() then
        Controls.FromTopSortEntryCheckbox:SetCheck(false);
    end
end
function OnFromTopSortEntryCheckbox()
    -- FromTopSortEntryCheckbox is tied to RepeatRouteCheckbox
    if Controls.FromTopSortEntryCheckbox:IsChecked() then
        Controls.RepeatRouteCheckbox:SetCheck(true);
    end
end
function RequestTradeRoute()
    local selectedUnit = UI.GetHeadSelectedUnit();
    if m_destinationCity and selectedUnit then
        local operationParams = {};
        operationParams[UnitOperationTypes.PARAM_X0] = m_destinationCity:GetX();
        operationParams[UnitOperationTypes.PARAM_Y0] = m_destinationCity:GetY();
```

```
        operationParams[UnitOperationTypes.PARAM_X1] = selectedUnit:GetX();
        operationParams[UnitOperationTypes.PARAM_Y1] = selectedUnit:GetY();
        if (UnitManager.CanStartOperation(selectedUnit,
UnitOperationTypes.MAKE_TRADE_ROUTE, nil, operationParams)) then
            UnitManager.RequestOperation(selectedUnit,
UnitOperationTypes.MAKE_TRADE_ROUTE, operationParams);
            UI.SetInterfaceMode(InterfaceModeTypes.SELECTION);
            UI.PlaySound("START_TRADE_ROUTE");
            -- Automated Handlers
            if Controls.RepeatRouteCheckbox:IsChecked() and
Controls.FromTopSortEntryCheckbox:IsChecked() then
                AutomateTrader(selectedUnit:GetID(), true, m_SortBySettings);
            elseif Controls.RepeatRouteCheckbox:IsChecked() then
                AutomateTrader(selectedUnit:GetID(), true);
            else
                AutomateTrader(selectedUnit:GetID(), false);
            end
        end
        return true;
    end
    return false;
end
-- ---------------------------------------------------------------------------
-- Sort bar insert buttons
-- ---------------------------------------------------------------------------
function OnGeneralSortBy(descArrowControl, sortByID)
    -- If shift is not being pressed, reset sort settings
    if not m_shiftDown then
        m_SortBySettings = {};
    end
    -- Sort based on currently showing icon toggled
    if descArrowControl:IsHidden() then
        InsertSortEntry(sortByID, SORT_DESCENDING, m_SortBySettings);
    else
        InsertSortEntry(sortByID, SORT_ASCENDING, m_SortBySettings);
    end
    m_SortSettingsChanged = true
    Refresh();
end
function OnSortByFood()
    OnGeneralSortBy(Controls.FoodDescArrow, SORT_BY_ID.FOOD)
end
function OnSortByProduction()
```

```
    OnGeneralSortBy(Controls.ProductionDescArrow, SORT_BY_ID.PRODUCTION)
end
function OnSortByGold()
    OnGeneralSortBy(Controls.GoldDescArrow, SORT_BY_ID.GOLD)
end
function OnSortByScience()
    OnGeneralSortBy(Controls.ScienceDescArrow, SORT_BY_ID.SCIENCE)
end
function OnSortByCulture()
    OnGeneralSortBy(Controls.CultureDescArrow, SORT_BY_ID.CULTURE)
end
function OnSortByFaith()
    OnGeneralSortBy(Controls.FaithDescArrow, SORT_BY_ID.FAITH)
end
function OnSortByTurnsToComplete()
    OnGeneralSortBy(Controls.TurnsToCompleteDescArrow,
SORT_BY_ID.TURNS_TO_COMPLETE)
end
-- ---------------------------------------------------------------------------
-- Sort bar delete buttons
-- ---------------------------------------------------------------------------
function OnGeneralNotSortBy(sortByID)
    RemoveSortEntry(sortByID, m_SortBySettings);
    m_SortSettingsChanged = true
    Refresh();
end
function OnNotSortByFood()
   OnGeneralNotSortBy(SORT_BY_ID.FOOD)
end
function OnNotSortByProduction()
    OnGeneralNotSortBy(SORT_BY_ID.PRODUCTION)
end
function OnNotSortByGold()
    OnGeneralNotSortBy(SORT_BY_ID.GOLD)
end
function OnNotSortByScience()
    OnGeneralNotSortBy(SORT_BY_ID.SCIENCE)
end
function OnNotSortByCulture()
    OnGeneralNotSortBy(SORT_BY_ID.CULTURE)
end
function OnNotSortByFaith()
    OnGeneralNotSortBy(SORT_BY_ID.FAITH)
```

```
end
function OnNotSortByTurnsToComplete()
    OnGeneralNotSortBy(SORT_BY_ID.TURNS_TO_COMPLETE)
end
-- ===========================================================================
--  Rise/Hide and refresh Trade UI
-- ===========================================================================
function OnInterfaceModeChanged( oldMode:number, newMode:number )
    if (oldMode == InterfaceModeTypes.MAKE_TRADE_ROUTE) then
        Close();
    end
    if (newMode == InterfaceModeTypes.MAKE_TRADE_ROUTE) then
        Open();
    end
end
function OnClose()
    Close();
    if UI.GetInterfaceMode() == InterfaceModeTypes.MAKE_TRADE_ROUTE then
        UI.SetInterfaceMode(InterfaceModeTypes.SELECTION);
    end
end
function Close()
    LuaEvents.TradeRouteChooser_SetTradeUnitStatus("");
    ContextPtr:SetHide(true);
    if UILens.IsLensActive(m_TradeRouteLens) then
        -- Make sure to switch back to default lens
        UILens.SetActive("Default");
    end
end
function Open()
    LuaEvents.TradeRouteChooser_SetTradeUnitStatus("LOC_HUD_UNIT_PANEL_CHOOSING_
TRADE_ROUTE");
    ContextPtr:SetHide(false);
    m_destinationCity = nil;
    Controls.RepeatRouteCheckbox:SetCheck(false);
    Controls.FromTopSortEntryCheckbox:SetCheck(false);
    -- Play Open Animation
    Controls.RouteChooserSlideAnim:SetToBeginning();
    Controls.RouteChooserSlideAnim:Play();
    -- Switch to TradeRoute Lens
    UILens.SetActive(m_TradeRouteLens);
    LuaEvents.TradeRouteChooser_Open();
    local selectedUnit:table = UI.GetHeadSelectedUnit();
```

```
    local selectedUnitID:number = selectedUnit:GetID();
    -- Select last route if one exists
    local lastRoute:table = GetLastRouteForTrader(selectedUnitID);
    if lastRoute ~= nil then
        print("Last route for trader " .. selectedUnitID .. ": " ..
GetTradeRouteString(lastRoute));
        originCity = Cities.GetCityInPlot(selectedUnit:GetX(),
selectedUnit:GetY());
        -- Don't select the route, if trader was transferred
        if lastRoute.OriginCityID ~= originCity:GetID() then
            print("Trader was transferred. Not selecting the last route")
        elseif IsRoutePossible(originCity:GetOwner(), originCity:GetID(),
lastRoute.DestinationCityPlayer, DestinationCityID) then
            local destinationPlayer:table =
Players[lastRoute.DestinationCityPlayer];
            m_destinationCity =
destinationPlayer:GetCities():FindID(lastRoute.DestinationCityID);
        else
            print("Route is no longer valid.");
        end
    else
        print("No last route was found for trader " .. selectedUnitID);
    end
    Refresh();
end
function CheckNeedsToOpen()
    if m_SkipNextOpen then
        m_SkipNextOpen = false
        return
    end
    local selectedUnit:table = UI.GetHeadSelectedUnit();
    if selectedUnit ~= nil then
        local selectedUnitInfo:table =
GameInfo.Units[selectedUnit:GetUnitType()];
        if selectedUnitInfo ~= nil and selectedUnitInfo.MakeTradeRoute == true
then
            local activityType:number =
UnitManager.GetActivityType(selectedUnit);
            if activityType == ActivityTypes.ACTIVITY_AWAKE and
selectedUnit:GetMovesRemaining() > 0 then
                -- If we're open and this is a trade unit then just refresh
                if not ContextPtr:IsHidden() then
                    Refresh();
```

```
                else
                    UI.SetInterfaceMode(InterfaceModeTypes.MAKE_TRADE_ROUTE);
                end
                -- Early out so we don't call Close()
                return;
            end
        end
    end
    -- If we're open and this unit is not a trade unit then close
    if not ContextPtr:IsHidden() then
        Close();
    end
end
function OnSkipNextOpen()
    m_SkipNextOpen = true
end
-- ===========================================================================
--  UI Events
-- ===========================================================================
function OnInit( isReload:boolean )
    if isReload then
        LuaEvents.GameDebug_GetValues( "TradeRouteChooser" );
    end
end
function OnShutdown()
    -- Cache values for hotloading...
    LuaEvents.GameDebug_AddValue("TradeRouteChooser", "filterIndex",
m_filterSelected );
    LuaEvents.GameDebug_AddValue("TradeRouteChooser", "destinationCity",
m_destinationCity );
end
-- ===========================================================================
--  LUA Event
--  Set cached values back after a hotload.
-- ===========================================================================s
function OnGameDebugReturn( context:string, contextTable:table )
    if context ~= "TradeRouteChooser" then
        return;
    end
    if contextTable["filterIndex"] ~= nil then
        m_filterSelected = contextTable["filterIndex"];
    end
    if contextTable["destinationCity"] ~= nil then
```

```
        m_destinationCity = contextTable["destinationCity"];
    end
    Refresh();
end
-- ===========================================================================
--  GAME Event
--  City was selected so close route chooser
-- ===========================================================================
function OnCitySelectionChanged(owner, ID, i, j, k, bSelected, bEditable)
    if not ContextPtr:IsHidden() and owner == Game.GetLocalPlayer() then
        OnClose();
    end
end
-- ===========================================================================
--  GAME Event
--  Unit was selected so close route chooser
-- ===========================================================================
function OnUnitSelectionChanged( playerID:number, unitID:number, hexI:number,
hexJ:number, hexK:number, bSelected:boolean, bEditable:boolean )
    -- Make sure we're the local player and not observing
    if playerID ~= Game.GetLocalPlayer() or playerID == -1 then
        return;
    end
    -- Don't call open/close if TradeOverview is open (needed to make
TradeOriginChooser open from TradeOverview)
    -- local pContextControl:table =
ContextPtr:LookUpControl("/InGame/TradeOverview");
    -- if pContextControl == nil then
    --     print("Cannot determine if partial screen \"/InGame/TradeOverview\"
is visible because it wasn't found at that path.");
    -- elseif not pContextControl:IsHidden() then
    --     print("Trade Overview Panel is open. Not opening Make Trade Route
screen.")
    --     return
    -- end
    -- If this is a de-selection event then close
    if not bSelected then
        OnClose();
        return;
    end
    CheckNeedsToOpen()
end
function OnLocalPlayerTurnEnd()
```

```
    if(GameConfiguration.IsHotseat()) then
        OnClose();
    end
    -- Clear cache to keep memory used low
    CacheEmpty()
end
function OnUnitActivityChanged( playerID :number, unitID :number, eActivityType
:number)
    -- Make sure we're the local player and not observing
    if playerID ~= Game.GetLocalPlayer() or playerID == -1 then
        return;
    end
    CheckNeedsToOpen();
end
function OnPolicyChanged( ePlayer )
    if not ContextPtr:IsHidden() and ePlayer == Game.GetLocalPlayer() then
        Refresh();
    end
end
-- ===========================================================================
--  Input
--  UI Event Handler
-- ===========================================================================
function KeyDownHandler( key:number )
    if key == Keys.VK_SHIFT then
        m_shiftDown = true;
        if not showSortOrdersPermanently then
            ShowSortOrderLabels();
        end
        -- let it fall through
    end
    return false;
end
function KeyUpHandler( key:number )
    if key == Keys.VK_SHIFT then
        m_shiftDown = false;
        if not showSortOrdersPermanently then
            HideSortOrderLabels();
        end
        -- let it fall through
    end
    if key == Keys.VK_RETURN then
        if m_destinationCity then
```

```
            RequestTradeRoute();
        end
        -- Dont let it fall through
        return true;
    end
    if key == Keys.VK_ESCAPE then
        OnClose();
        return true;
    end
    return false;
end
function OnInputHandler( pInputStruct:table )
    local uiMsg = pInputStruct:GetMessageType();
    if uiMsg == KeyEvents.KeyDown then return KeyDownHandler(
pInputStruct:GetKey() ); end
    if uiMsg == KeyEvents.KeyUp then return KeyUpHandler( pInputStruct:GetKey()
); end
    return false;
end
-- ===========================================================================
function OnSelectRouteFromOverview( destinationOwnerID:number,
destinationCityID:number )
    m_postOpenSelectPlayerID = destinationOwnerID;
    m_postOpenSelectCityID = destinationCityID;
    CheckNeedsToOpen()
end
-- ===========================================================================
--  Setup
-- ===========================================================================
function InitButton(control, callbackLClick, callbackRClick)
    control:RegisterCallback(Mouse.eLClick, callbackLClick)
    if callbackRClick ~= nil then
        control:RegisterCallback(Mouse.eRClick, callbackRClick)
    end
    control:RegisterCallback( Mouse.eMouseEnter, function()
UI.PlaySound("Main_Menu_Mouse_Over") end)
end
function Initialize()
    print("Initializing BTS Trade Route Chooser");
    TradeSupportAutomater_Initialize();
    -- Context Events
    ContextPtr:SetInitHandler( OnInit );
    ContextPtr:SetShutdown( OnShutdown );
```

```
    ContextPtr:SetInputHandler( OnInputHandler, true );
    -- Lua Events
    LuaEvents.GameDebug_Return.Add( OnGameDebugReturn );
    -- Context Events
    LuaEvents.TradeRouteChooser_SkipOpen.Add( OnSkipNextOpen )
    LuaEvents.TradeOverview_SelectRouteFromOverview.Add(
OnSelectRouteFromOverview );
    LuaEvents.TradeRouteChooser_Close.Add( OnClose )
    -- Game Engine Events
    Events.InterfaceModeChanged.Add( OnInterfaceModeChanged );
    Events.CitySelectionChanged.Add( OnCitySelectionChanged );
    Events.UnitSelectionChanged.Add( OnUnitSelectionChanged );
    Events.UnitActivityChanged.Add( OnUnitActivityChanged );
    Events.LocalPlayerTurnEnd.Add( OnLocalPlayerTurnEnd );
    Events.GovernmentPolicyChanged.Add( OnPolicyChanged );
    Events.GovernmentPolicyObsoleted.Add( OnPolicyChanged );
    -- Control Events
    InitButton(Controls.BeginRouteButton, RequestTradeRoute)
    InitButton(Controls.Header_CloseButton, OnClose )
    -- Filter
    Controls.FilterButton:RegisterCallback( Mouse.eLClick, UpdateFilterArrow );
    Controls.DestinationFilterPulldown:RegisterSelectionCallback(
OnFilterSelected );
    -- Control events - checkboxes
    InitButton(Controls.RepeatRouteCheckbox, OnRepeatRouteCheckbox );
    InitButton(Controls.FromTopSortEntryCheckbox, OnFromTopSortEntryCheckbox );
    -- Control events - sort bar
    InitButton(Controls.FoodSortButton, OnSortByFood, OnNotSortByFood)
    InitButton(Controls.ProductionSortButton, OnSortByProduction,
OnNotSortByProduction)
    InitButton(Controls.GoldSortButton, OnSortByGold, OnNotSortByGold)
    InitButton(Controls.ScienceSortButton, OnSortByScience, OnNotSortByScience)
    InitButton(Controls.CultureSortButton, OnSortByCulture, OnNotSortByCulture)
    InitButton(Controls.FaithSortButton, OnSortByFaith, OnNotSortByFaith)
    InitButton(Controls.TurnsToCompleteSortButton, OnSortByTurnsToComplete,
OnNotSortByTurnsToComplete)
end
Initialize();
```

```
<?xml version="1.0" encoding="utf-8"?>
<Context xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSc
hemaLocation="..\..\..\..\..\CivTech\Libs\ForgeUI\ForgeUI_Assets\Controls.xsd"
Style="FontNormal20">
  <!-- Left Panel -->
  <SlideAnim ID="RouteChooserSlideAnim" Anchor="L,T" Begin="-350,0" End="0,0"
Cycle="Once" Speed="2" Function="OutQuint">
    <Container ID="RouteChooser" Offset="-3,54">
      <!-- Body container-->
      <Container Size="395,parent" ConsumeAllMouse="1">
        <Stack StackGrowth="Down">
          <!-- Top Panel -->
          <Grid ID="TopGrid" Size="parent-4,auto" MinSize="0,240"
Texture="DestinationChooser_CurrentSlot" SliceStart="0,0" SliceCorner="30,30"
SliceSize="200,60" SliceTextureSize="308,173">
            <Container ID="CurrentSelectionContainer" Size="parent, auto"
Offset="0,7">
              <Stack Anchor="C,T" StackGrowth="Down">
                <!-- Selected City -->
                <Grid ID="BannerBase" Anchor="C,T" Offset="0,12"
Size="parent-70,33" Texture="CityPanel_BannerBase" SliceCorner="20,10"
SliceSize="160,1" SliceTextureSize="199,33" Color="150,170,100,255">
                  <Grid ID="BannerDarker" Anchor="L,T" Offset="4,2"
Size="parent-8,parent-10" Texture="CityPanel_BannerDarker" SliceCorner="95,10"
SliceSize="1,1" SliceTextureSize="191,23" Color="0,0,0,100" />
                  <Grid ID="BannerLighter" Anchor="L,T" Offset="4,2"
Size="parent-8,parent-10" Texture="CityPanel_BannerLighter" SliceCorner="95,10"
SliceSize="1,1" SliceTextureSize="191,23" Color="255,255,255,255" />
                  <Grid Anchor="L,T" Offset="6,2" Size="parent-10,parent-8"
Texture="CityPanel_BannerNone" SliceCorner="70,10" SliceSize="1,1"
SliceTextureSize="179,20" Color="255,0,0,255" />
                  <Label ID="CityName" Anchor="L,C" Offset="40,-2"
Style="FontFlair16" FontStyle="Stroke" EffectColor="0,0,0,25"
String="$CityName$" SmallCaps="20" SmallCapsType="EveryWord" TruncateWidth="220"
/>
                  <Label ID="TradingPostIcon" Anchor="L,C" Offset="14,-4"
Style="FontNormal16" FontStyle="Shadow" String="[ICON_TradingPost]"
Color="255,0,0,255" EffectColor="255,0,0,255" GradientColor="255,0,0,255" />
                  <Label ID="CityStateQuestIcon" Anchor="L,C" Offset="-2,-12"
Style="FontNormal16" FontStyle="Shadow" String="[ICON_CityStateQuest]"
Color="255,0,0,255" EffectColor="255,0,0,255" GradientColor="255,0,0,255" />
                  <Stack StackGrowth="Left" Offset="14,-4" Anchor="R,C">
                    <Label ID="TurnsToComplete" Anchor="R,C" Style="FontFlair16"
```

```
String="??" FontStyle="Stroke" EffectColor="0,0,0,25" />
                    <Label ID="TurnsToCompleteIcon"  String="[ICON_Turn]"
Offset="0,-1" Anchor="R,C" Style="FontFlair16" FontStyle="Stroke"
EffectColor="0,0,0,25" />
                  </Stack>
                </Grid>
                <!-- Resource Information -->
                <Grid ID="DestinationResources" Size="parent-25,35" Anchor="C,T"
Offset="1,16" Texture="Controls_ItemContainerDestination" SliceCorner="6,2"
SliceSize="2,31" SliceTextureSize="24,35" >
                  <!-- Column Background -->
                  <Box Size="50,parent" Offset="50,0" Color="0,0,0,22"/>
                  <Box Size="50,parent" Offset="150,0" Color="0,0,0,22"/>
                  <Box Size="50,parent" Offset="250,0" Color="0,0,0,22"/>
                  <!-- <Box Size="50,parent" Offset="200,0" Color="0,0,0,22"/>
-->
                  <Label ID="DestinationResourceHeader" Size="300,20"
Anchor="L,T" Offset="2,-15" Style="FontNormal14" Color="0,0,0,150"
TruncateWidth="279"/>
                  <Stack ID="DestinationResourceList" Anchor="L,T" Offset="3,8"
StackGrowth="Bottom"/>
                  <Label ID="DestinationReceivesNoBenefitsLabel" Anchor="C,C"
Offset="-4,0" Style="FontNormal16" FontStyle="Shadow" EffectColor="Black"
String="LOC_ROUTECHOOSER_NO_BENEFITS_FROM_ROUTE" Hidden="1"/>
                </Grid>
                <Grid ID="OriginResources" Size="parent-25,35" Anchor="R,T"
Offset="2,24" Texture="Controls_ItemContainerOrigin" SliceCorner="14,2"
SliceSize="3,31" SliceTextureSize="24,35" >
                  <!-- Column Background -->
                  <Box Size="50,parent" Offset="53,0" Color="0,0,0,22"/>
                  <Box Size="50,parent" Offset="153,0" Color="0,0,0,22"/>
                  <Box Size="50,parent" Offset="253,0" Color="0,0,0,22"/>
                  <Label ID="OriginResourceHeader" Size="250,20" Anchor="L,T"
Offset="13,-15" Style="FontNormal14" Color="0,0,0,150" TruncateWidth="279"/>
                  <Stack ID="OriginResourceList" Anchor="L,C" Offset="6,-1"
StackGrowth="Bottom"/>
                  <Label ID="OriginReceivesNoBenefitsLabel" Anchor="C,C"
Style="FontNormal16" FontStyle="Shadow" EffectColor="Black"
String="LOC_ROUTECHOOSER_NO_BENEFITS_FROM_ROUTE" Hidden="1"/>
                </Grid>
                <Stack StackGrowth="Right" Anchor="C,B" Offset="0,6"
StackPadding="5">
                  <Label Offset="0,2" FontStyle="Shadow" WrapWidth="140"
```

```
Style="FontNormal14" String="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_LABEL"
ToolTip="LOC_TRADE_REPEAT_ROUTE_CHECKBOX_TOOLTIP" Color="White"/>
                  <CheckBox ID="RepeatRouteCheckbox" Style="MainCheckBox"
IsChecked="0"/>
                  <Label Offset="20,2" FontStyle="Shadow" WrapWidth="140"
Style="FontNormal14" String="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_LABEL"
ToolTip="LOC_TRADE_FROM_TOP_SORT_ENTRY_CHECKBOX_TOOLTIP" Color="White"/>
                  <CheckBox ID="FromTopSortEntryCheckbox" Style="MainCheckBox"
IsChecked="0"/>
                </Stack>
                <!-- Confirm Button -->
                <GridButton ID="BeginRouteButton" Size="parent-100,26"
Offset="0,7" Anchor="C,B" String="LOC_ROUTECHOOSER_BEGIN_ROUTE_BUTTON"
Style="FontFlair14" FontStyle="Shadows" TextOffset="0,2" EffectColor="0,0,0,255"
SmallCaps="20" SmallCapsType="EveryWord">
                  <GridData Texture="Controls_ConfirmSmall"
StateOffsetIncrement="0,26" SliceCorner="26,13" SliceSize="2,2"
SliceTextureSize="51,26"/>
                </GridButton>
              </Stack>
            </Container>
            <Container Size="parent, auto" MinSize="0,240">
              <Label ID="StatusMessage" Size="300,50" Anchor="C,C"
Style="FontNormal16" Color="0,0,0,150"/>
            </Container>
          </Grid>
          <!-- Bottom Panel -->
          <Grid Size="parent,parent-296" Anchor="L,T" Offset="-7,-1"
Texture="Controls_ContainerBlue" SliceStart="0,0" SliceCorner="3,3"
SliceSize="9,9" SliceTextureSize="16,16">
            <!-- Filter Pulldown -->
            <PullDown ID="DestinationFilterPulldown" ConsumeMouse="0"
Offset="5,28" Anchor="C,T" Size="360,30" AutoSizePopUp="1" AutoFlip="1"
ScrollThreshold="400">
              <ButtonData>
                <GridButton   ID="FilterButton" TextAnchor="R,C"
TextOffset="15,0" Style="FontNormal14" FontStyle="Shadow"
EffectColor="0,0,0,255" Offset="0,-25" Size="50,-20"
Texture="Controls_ButtonControl.dds" SliceCorner="10,10" SliceSize="1,1"
SliceTextureSize="24,24" />
              </ButtonData>
              <GridData InnerPadding="15,15" Offset="0,0" Anchor="L,T"
Style="Drawer"/>
```

```
              <ScrollPanelData Anchor="L,T"  Vertical="1" Size="11,14"
Offset="0,0" AutoScrollBar="1">
                <ScrollBar Style="ScrollVerticalBacking" Anchor="L,T"
AnchorSide="I,I" Color="28,60,90,255" Offset="-2,2">
                  <Thumb Style="ScrollThumbAlt" Color="28,60,90,255" />
                </ScrollBar>
              </ScrollPanelData>
              <StackData StackGrowth="Bottom" Offset="0,0" Size="200,400"
Anchor="L,T" />
              <InstanceData Name="FilterEntry">
                <GridButton Anchor="L,T" ID="Button" Size="265,26" Offset="1,0"
Style="FontNormal14" FontStyle="Shadow" EffectColor="0,0,0,255"
Texture="Controls_ButtonControl.dds" SliceCorner="10,10" SliceSize="1,1"
SliceTextureSize="24,24" StateOffsetIncrement="0,24"/>
              </InstanceData>
              <!-- Show Route Text -->
              <Label String="LOC_ROUTECHOOSER_FILTER_SHOWROUTES" Anchor="L,T"
Offset="30,-17" Style="FontNormal14" FontStyle="Shadow"
EffectColor="0,0,0,255"/>
              <Image ID="PulldownOpenedArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,0" Size="20,16"
Anchor="L,T" Offset="7,-18"/>
              <Image ID="PulldownClosedArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,60" Size="20,16"
Anchor="L,T" Offset="7,-22"/>
            </PullDown>
            <!-- Sort Yield Stack -->
            <Container ID="RouteContainer" Anchor="L,T" Offset="17,0">
              <Stack Anchor="L,T" ID="SortBarStack" StackGrowth="Right"
Offset="11,34" StackPadding="2">
                <GridButton ID="FoodSortButton"
ToolTip="LOC_TRADE_SORT_BY_FOOD_TOOLTIP" Offset="0,0" Size="48,18"
Color="255,255,255,255" Style="PanelButtonLightweight">
                  <Label ID="FoodSortLabel" Anchor="R,C" Offset="-2,1"
Style="FontNormal12" String="[Icon_Food]"/>
                  <Label ID="FoodSortOrder" Hidden="1" Anchor="C,C" Offset="0,0"
String="9" Style="FontNormal12"/>
                  <Image ID="FoodDescArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,0" Size="20,16"
Anchor="L,C" Offset="2,0"/>
                  <Image ID="FoodAscArrow" Texture="Controls_ButtonExtendSmall2"
TextureOffset="0,60" Size="20,16" Anchor="L,C" Offset="2,-5"/>
                </GridButton>
```

```
                <GridButton ID="ProductionSortButton"
ToolTip="LOC_TRADE_SORT_BY_PRODUCTION_TOOLTIP" Size="48,18"
Color="255,255,255,255" Style="PanelButtonLightweight">
                  <Label ID="ProductionSortLabel" Anchor="R,C" Offset="-2,1"
Style="FontNormal12" String="[Icon_Production]"/>
                  <Label ID="ProductionSortOrder" Hidden="1" Anchor="C,C"
Offset="0,0" String="9" Style="FontNormal12"/>
                  <Image ID="ProductionDescArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,0" Size="20,16"
Anchor="L,C" Offset="2,0"/>
                  <Image ID="ProductionAscArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,60" Size="20,16"
Anchor="L,C" Offset="2,-5"/>
                </GridButton>
                <GridButton ID="GoldSortButton"
ToolTip="LOC_TRADE_SORT_BY_GOLD_TOOLTIP" Size="48,18" Color="255,255,255,255"
Style="PanelButtonLightweight">
                  <Label ID="GoldSortLabel" Anchor="R,C" Offset="-2,1"
Style="FontNormal12" String="[Icon_Gold]"/>
                  <Label ID="GoldSortOrder" Hidden="1" Anchor="C,C" Offset="0,0"
String="9" Style="FontNormal12"/>
                  <Image ID="GoldDescArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,0" Size="20,16"
Anchor="L,C" Offset="2,0"/>
                  <Image ID="GoldAscArrow" Texture="Controls_ButtonExtendSmall2"
TextureOffset="0,60" Size="20,16" Anchor="L,C" Offset="2,-5"/>
                </GridButton>
                <GridButton ID="ScienceSortButton"
ToolTip="LOC_TRADE_SORT_BY_SCIENCE_TOOLTIP" Size="48,18" Color="255,255,255,255"
Style="PanelButtonLightweight">
                  <Label ID="ScienceSortLabel" Anchor="R,C" Offset="-2,1"
Style="FontNormal12" String="[Icon_Science]"/>
                  <Label ID="ScienceSortOrder" Hidden="1" Anchor="C,C"
Offset="0,0" String="9" Style="FontNormal12"/>
                  <Image ID="ScienceDescArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,0" Size="20,16"
Anchor="L,C" Offset="2,0"/>
                  <Image ID="ScienceAscArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,60" Size="20,16"
Anchor="L,C" Offset="2,-5"/>
                </GridButton>
                <GridButton ID="CultureSortButton"
ToolTip="LOC_TRADE_SORT_BY_CULTURE_TOOLTIP" Size="48,18" Color="255,255,255,255"
```

```
Style="PanelButtonLightweight">
                  <Label ID="CultureSortLabel" Anchor="R,C" Offset="-2,1"
Style="FontNormal12" String="[Icon_Culture]"/>
                  <Label ID="CultureSortOrder" Hidden="1" Anchor="C,C"
Offset="0,0" String="9" Style="FontNormal12"/>
                  <Image ID="CultureDescArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,0" Size="20,16"
Anchor="L,C" Offset="2,0"/>
                  <Image ID="CultureAscArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,60" Size="20,16"
Anchor="L,C" Offset="2,-5"/>
                </GridButton>
                <GridButton ID="FaithSortButton"
ToolTip="LOC_TRADE_SORT_BY_FAITH_TOOLTIP" Size="48,18" Color="255,255,255,255"
Style="PanelButtonLightweight">
                  <Label ID="FaithSortLabel" Anchor="R,C" Offset="-2,1"
Style="FontNormal12" String="[Icon_Faith]"/>
                  <Label ID="FaithSortOrder" Hidden="1" Anchor="C,C"
Offset="0,0" String="9" Style="FontNormal12"/>
                  <Image ID="FaithDescArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,0" Size="20,16"
Anchor="L,C" Offset="2,0"/>
                  <Image ID="FaithAscArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,60" Size="20,16"
Anchor="L,C" Offset="2,-5"/>
                </GridButton>
                <GridButton ID="TurnsToCompleteSortButton"
ToolTip="LOC_TRADE_SORT_BY_TURNS_REMAINING_TOOLTIP" Size="48,18"
Color="255,255,255,255" Style="PanelButtonLightweight">
                  <Label ID="TurnsToCompleteSortLabel" Anchor="R,C"
Offset="-2,0" Style="FontNormal14" String="[Icon_Turn]"/>
                  <Label ID="TurnsToCompleteSortOrder" Hidden="1" Anchor="C,C"
Offset="0,0" String="9" Style="FontNormal12"/>
                  <Image ID="TurnsToCompleteDescArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,0" Size="20,16"
Anchor="L,C" Offset="2,0"/>
                  <Image ID="TurnsToCompleteAscArrow"
Texture="Controls_ButtonExtendSmall2" TextureOffset="0,60" Size="20,16"
Anchor="L,C" Offset="2,-5"/>
                </GridButton>
              </Stack>
              <!-- Destination List -->
              <ScrollPanel ID="RouteChoiceScrollPanel" Anchor="L,T"
```

```
Offset="0,54" Size="parent,parent-54" Vertical="1" AutoScrollBar="1"
AutoSizeScrollBar="1">
                <ScrollBar Style="ScrollVerticalBacking" Anchor="L,T"
Color="28,60,90,255" Offset="-6,2">
                  <Thumb Style="ScrollThumbAlt" Color="28,60,90,255" />
                </ScrollBar>
                <!-- Destination Stack -->
                <Stack ID="RouteChoiceStack" Anchor="L,T" Offset="5,0"
StackGrowth="Down"/>
              </ScrollPanel>
            </Container>
            <!-- Bottom Divider -->
            <Grid Size="parent+5,10" Anchor="C,B" Offset="0,-8"
Style="Divider3Grid" />
          </Grid>
        </Stack>
      </Container>
      <!-- Header container-->
      <Grid ID="HeaderBannerGrid" Size="400,60" Offset="3,-25"
Style="HeaderBannerLeft" ConsumeAllMouse="1">
        <Label ID="Header_OriginText" Size="200, 50" Offset="0,7" Anchor="C,T"
Style="FontFlair16" Color0="0,59,77,255" SmallCaps="20"
SmallCapsType="FirstOnly" String="$Origin To...$"/>
        <Button ID="Header_CloseButton" Size="32,32" Offset="10,-2" Anchor="R,T"
Texture="Controls_CloseButtonAlt"/>
      </Grid>
    <Tutorial  ID="TutTradePanel" Style="TutorialContainer" AnchorSide="O,I"
Offset="-280,190" TriggerBy="TutorialTradePanel" >
      <SlideAnim Anchor="C,T" Start="30,0" EndOffset="40,0" Cycle="Bounce"
Function="OutQuad" >
        <Image Texture="Tutorial_ArrowH" Offset="0,0" Size="58,44" FlipX="1"/>
      </SlideAnim>
    </Tutorial>
    <Tutorial  ID="TutTradeRoute" Style="TutorialContainer" AnchorSide="O,I"
Offset="-280,250" TriggerBy="TutorialTradeRoute" >
      <Grid Style="TutorialCalloutGrid" Offset="220,0">
        <Label Style="TutorialBodyText" String="LOC_META_145b_BODY" />
      </Grid>
      <SlideAnim Anchor="C,T" Start="30,0" EndOffset="40,0" Cycle="Bounce"
Function="OutQuad" >
        <Image Texture="Tutorial_ArrowH" Offset="0,0" Size="58,44" FlipX="1"/>
      </SlideAnim>
    </Tutorial>
```

```
    <Tutorial  ID="TutBeginRoute" Style="TutorialContainer" AnchorSide="O,I"
Offset="-260,154" TriggerBy="TutorialBeginRoute" >
      <Grid Style="TutorialCalloutGrid" Offset="220,0">
        <Label Style="TutorialBodyText" String="LOC_META_145c_BODY" />
      </Grid>
      <SlideAnim Anchor="C,T" Start="30,0" EndOffset="40,0" Cycle="Bounce"
Function="OutQuad" >
        <Image Texture="Tutorial_ArrowH" Offset="0,0" Size="58,44" FlipX="1"/>
      </SlideAnim>
    </Tutorial>
    </Container>
  </SlideAnim>
  <!-- ======================= -->
  <!--        Instances        -->
  <!-- ======================= -->
  <Instance       Name="RouteChoiceInstance">
    <Container ID="Top" Size="375,110" Offset="0,2">
      <GridButton ID="Button" Size="parent,parent" Anchor="C,T">
        <GridData Texture="DestinationChooser_Button"
StateOffsetIncrement="0,76" SliceCorner="14,14" SliceSize="19,44"
SliceTextureSize="48,76"/>
        <!-- Selector Brace -->
        <Grid ID="SelectorBrace" Size="parent+4,parent+4" Anchor="C,C"
Texture="Controls_SelectorBrace" SliceCorner="29,29" SliceSize="2,2"
SliceTextureSize="62,62"/>
        <!-- City Banner -->
        <Grid               ID="BannerBase"               Anchor="C,T"
Offset="-1,4"    Size="parent-10,33"
Texture="CityPanel_BannerBase"        SliceCorner="20,10" SliceSize="160,1"
SliceTextureSize="199,33" Color="150,170,100,255">
          <Grid             ID="BannerDarker"
Anchor="L,T" Offset="4,2"     Size="parent-8,parent-10"
Texture="CityPanel_BannerDarker"      SliceCorner="95,10" SliceSize="1,1"
SliceTextureSize="191,23" Color="0,0,0,100" />
          <Grid             ID="BannerLighter"
Anchor="L,T" Offset="4,2"     Size="parent-8,parent-10"
Texture="CityPanel_BannerLighter"     SliceCorner="95,10" SliceSize="1,1"
SliceTextureSize="191,23" Color="255,255,255,255" />
          <Grid                                       Anchor="L,T" Offset="6,2"
Size="parent-10,parent-8"   Texture="CityPanel_BannerNone"
SliceCorner="70,10" SliceSize="1,1" SliceTextureSize="179,20"
Color="255,0,0,255" />
          <Label            ID="CityName"             Anchor="L,C"
```

```
Offset="34,-2"   Style="FontFlair16"         FontStyle="Stroke"
EffectColor="0,0,0,25" String="$CityName$" SmallCaps="20"
SmallCapsType="EveryWord" TruncateWidth="225"/>
          <Label            ID="TradingPostIcon"  Anchor="L,C" Offset="10,-4"
Style="FontNormal16" FontStyle="Shadow" String="[ICON_TradingPost]"
Color="255,0,0,255" EffectColor="255,0,0,255" GradientColor="255,0,0,255" />
          <Label            ID="CityStateQuestIcon" Anchor="L,C" Offset="-2,-12"
Style="FontNormal16" FontStyle="Shadow" String="[ICON_CityStateQuest]"
Color="255,0,0,255" EffectColor="255,0,0,255" GradientColor="255,0,0,255" />
          <Stack StackGrowth="Left" Anchor="R,C" Offset="12,-4"
StackPadding="0">
            <Label ID="TurnsToComplete" Anchor="C,C" Offset="0,0"
Style="FontFlair14" FontStyle="Stroke" EffectColor="0,0,0,25" String="??" />
            <Label ID="TurnsToCompleteIcon"  String="[ICON_Turn]" Anchor="C,C"
Offset="0,-1"  Style="FontFlair16" FontStyle="Stroke" EffectColor="0,0,0,25" />
          </Stack>
        </Grid>
        <!-- Resource Info -->
        <Grid Size="parent-13,parent-45" Anchor="C,B" Offset="-1,7"
Texture="Controls_ItemContainer" SliceCorner="8,8" SliceTextureSize="16,16" >
          <!-- Column Background -->
          <Box Size="50,parent" Offset="50,0" Color="0,0,0,22"/>
          <Box Size="50,parent" Offset="150,0" Color="0,0,0,22"/>
          <Box Size="50,parent" Offset="250,0" Color="0,0,0,22"/>
          <!-- <Box Size="50,parent" Offset="350,0" Color="0,0,0,22"/> -->
          <Stack ID="ResourceList" Anchor="L,C" Offset="4,0"
StackGrowth="Bottom" StackPadding="10"/>
        </Grid>
      </GridButton>
    </Container>
  </Instance>
  <!-- Route Yields Instance -->
  <Instance Name="RouteYieldInstance">
    <Stack ID="YieldStack" StackGrowth="Right" StackPadding="2">
      <Container Size="48,20">
        <Label ID="YieldFoodLabel" Color="Food" Anchor="C,C" Offset="0,0"
Style="FontNormal12" FontStyle="Shadow" String="+15[Icon_Food]"/>
      </Container>
      <Container Size="48,20">
        <Label ID="YieldProductionLabel" Color="Production" Anchor="C,C"
Offset="0,0" Style="FontNormal12" FontStyle="Shadow"
String="+15[Icon_Production]"/>
      </Container>
```

```
      <Container Size="48,20">
        <Label ID="YieldGoldLabel" Color="Gold" Anchor="C,C" Offset="0,0"
Style="FontNormal12" FontStyle="Shadow" String="+15[Icon_Gold]"/>
      </Container>
      <Container Size="48,20">
        <Label ID="YieldScienceLabel" Color="Science" Anchor="C,C" Offset="0,0"
Style="FontNormal12" FontStyle="Shadow" String="+15[Icon_Science]"/>
      </Container>
      <Container Size="48,20">
        <Label ID="YieldCultureLabel" Color="Culture" Anchor="C,C" Offset="0,0"
Style="FontNormal12" FontStyle="Shadow" String="+15[Icon_Culture]"/>
      </Container>
      <Container Size="48,20">
        <Label ID="YieldFaithLabel" Color="Faith" Anchor="C,C" Offset="0,0"
Style="FontNormal12" FontStyle="Shadow" String="+15[Icon_Faith]"/>
      </Container>
      <Container ID="RouteReligionContainer" Size="56,20" Hidden="1">
        <Container Anchor="C,C" Offset="0,0" >
          <Image ID="ReligionIconBacking" Anchor="R,C"
Texture="Religion_FollowersSlot" Size="30,30">
            <Image ID="ReligionIcon" Anchor="C,C" Size="22,22" IconSize="22"/>
          </Image>
          <Label ID="ResourceEntryText" Anchor="L,C" Size="50,10" Offset="2,0"
String="+0.5" Style="FontNormal12"/>
        </Container>
      </Container>
    </Stack>
  </Instance>
</Context>
```