SCIENTIFIC REGISTRY OF TRANSPLANT RECIPIENTS

Kidney-Pancreas Simulated Allocation Model

KP SAM

Version 2015

User's Guide

Last Updated: April 20, 2015

This work is supported by the Scientific Registry of Transplant Recipients (contract number HHSH250201000018C), with funding and oversight by the Health Resources and Services Administration, an agency of the U.S. Department of Health and Human Services. The Scientific Registry of Transplant Recipients is operated by the Minneapolis Medical Research Foundation, Chronic Disease Research Group.

Contributors to SAM refinement and redesign

Minneapolis Medical Research Foundation
Yi Peng
Eugene Shteyn
Xinyue Wang
Josh Pyke

University of Minnesota
Diwakar Gupta
Hao-Wei Chen
Zhi Zhang
Sang Phil Kim

United States Naval Academy
Sommer Gentry

Johns Hopkins University
Dorry Segev

Contributors to initial SAM development

Arbor Research Collaborative for Health
David Dickinson
Shannon Li
Keith McCullough
Robert M. Merion, MD
Friedrich K. Port, MD
Ann M. Rodgers
Caroline Shevrin
Robert A. Wolfe

University of Michigan
Cora Brunton
Susan Murray

Altarum Institute
David Thompson
Larry Waisanen

SCIENTIFIC REGISTRY OF TRANSPLANT RECIPIENTS

Table of Contents

1 Getting Started

1.1 Purpose of the Program

The purpose of the Kidney-Pancreas Simulated Allocation Model (KPSAM) is to simulate the allocation of kidneys and/or pancreata to candidates waiting for organ transplants and their outcomes. Input files are provided based upon the OPTN waiting list and organs. KPSAM is a computer simulation program developed by the Scientific Registry of Transplant Recipients (SRTR).

The program has been designed to support studies of alternative organ allocation policies. It can use a variety of allocation rules to determine how a series of organs would be allocated to a list of potential recipients under each of the rules considered. The allocation process involves some random components reflecting the uncertainty in acceptance decisions when an organ is offered to a potential recipient and reflecting the unpredictable life expectancy that can result from receiving a transplant or not. In order to account for such random variation, the program can make organ allocations several times with the same set of allocation rules, candidate lists, and organs in order to determine what happens on average.

This guide describes the means by which the user can simulate the organ allocation process by creating or editing an Allocation Run package, which consists of a named collection of input files and specifications. Default files that reflect actual experience are provided for this purpose. Alternatively, input files could be created by the user with candidate characteristics, candidate histories, and organ characteristics.

1.2 Installing the Software

You must be running Microsoft Windows XP, Windows Vista or Windows 7 to install the Kidney-Pancreas Simulated Allocation Model. You may need administrator rights to install KPSAM. The software for the current version of the KPSAM program is available on CD from the SRTR.

1. Insert the software CD into your CD drive.
2. Using Windows Explorer, select your CD drive and double-click on setup.exe. The Setup Wizard will then start.
3. The Setup Wizard will prompt you to select Next to continue installation.
4. After reading the Information screen, please select Next.
5. You may type in your name and company, then select Next.
6. The setup program will show the recommended default folder for installing the program, which can be changed. Select Next.

7. The setup program will ask you if you want to create a start menu folder and desktop shortcut
8. The Setup Wizard gives you a summary of your installation settings. Select Install to continue.
9. Software program files will be copied to the appropriate folders. A folder called \SRTR\KPSAM will be created on your hard disk, and sample data files will be put into \SRTR\KPSAM\Input\.
10. Select Finish to exit the Setup Wizard Completed screen. The process should take less than two minutes. Remove the CD from your drive and store in CD case.

1.3 Deleting the Software

If you choose to remove the program from your computer, do not start by deleting the files in the KPSAM folder. On the taskbar at the bottom left of your screen, click the Start button to bring up the Start menu. Choose Settings, then Control Panel. Double-click on the icon Add/Remove Programs. Then highlight Liver Simulated Allocation Model, click Add/Remove, and follow the on-screen instructions.

You may also select the uninstall option from the start menu folder if you chose to create one during installation.

1.4 Moving the Software

If you want to move the software, first delete it as described above, then reinstall the software to a new location.

1.5 Starting the Program

The Setup Wizard has created two shortcuts that you can use to start the KPSAM program:

On the taskbar at the bottom-left of your screen, click the Start button; the Start menu appears. Choose Programs, SRTR, then KPSAM.

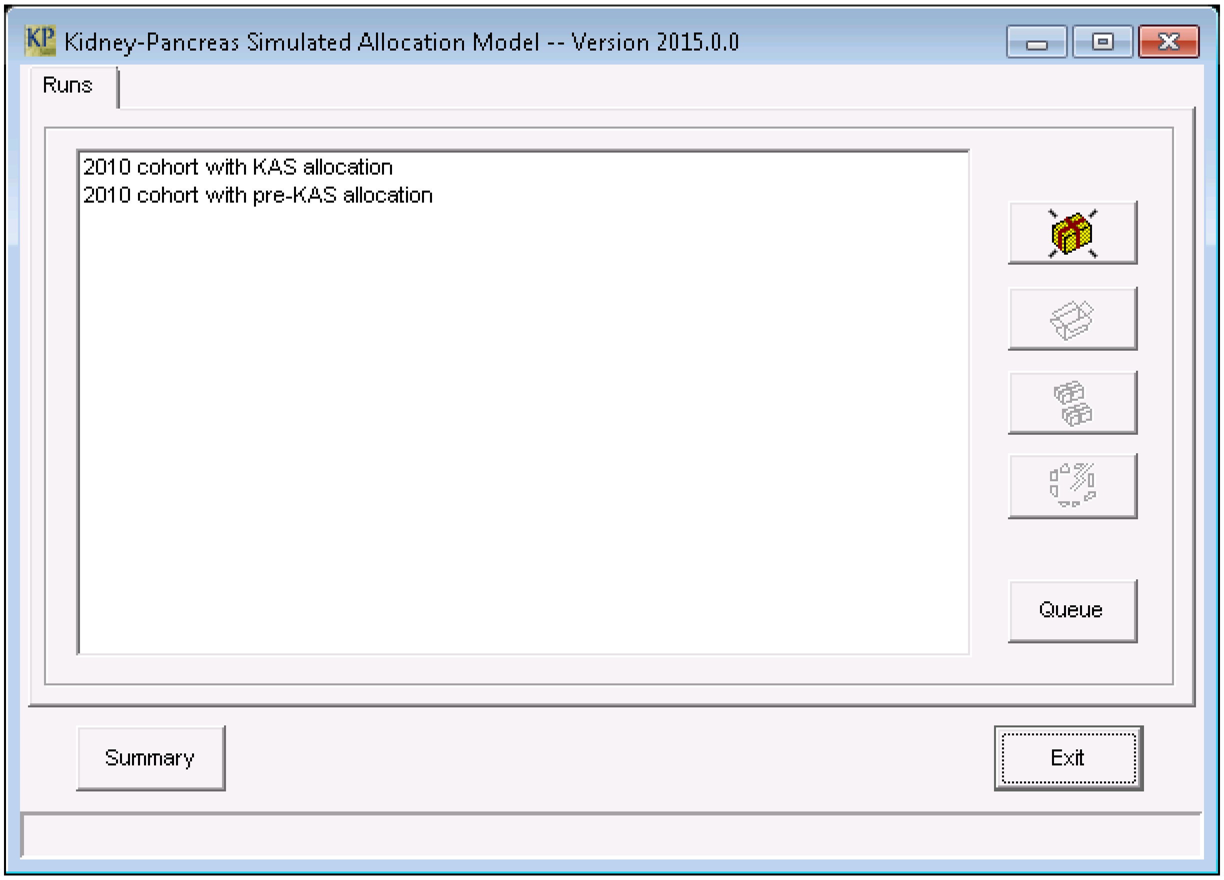
Alternatively, you can click on the KPSAM icon that the Setup Wizard put on your Desktop.

You may also open the folder the program was installed in and click on the KPSAM icon.

1.6 Setting Input Parameters

The information that KPSAM uses must be prepared according to very detailed specifications. This information is extensive, and computer files must be prepared with this information. Descriptions of these files are available in Chapters 5 and 6 of this document. After you have prepared these files, you can use them with KPSAM. The KPSAM user interface prompts users to identify the files that contain the input data (organ and candidate characteristics and other information related to organ allocation, acceptance, and post-transplant survival) that define a single Allocation Run of the program. The interface also supports the direct input of selected aspects of the allocation rules in dialogue boxes in the program.

The next few pages will take you through the steps needed to create an Allocation Run Package. Below is the first screen of the program. To get started, click on the Package button on the right of the screen.


Kidney-Pancreas Simulated Allocation Model -- Version 2015.0.0
Runs
Queue
Summary
Exit

2010 cohort with KAS allocation
2010 cohort with pre-KAS allocation

1.6.1 Run Specification

The Run Specification screen contains two tabs, which are used to select the files from which the program will read its input data for this Allocation Run Specification.

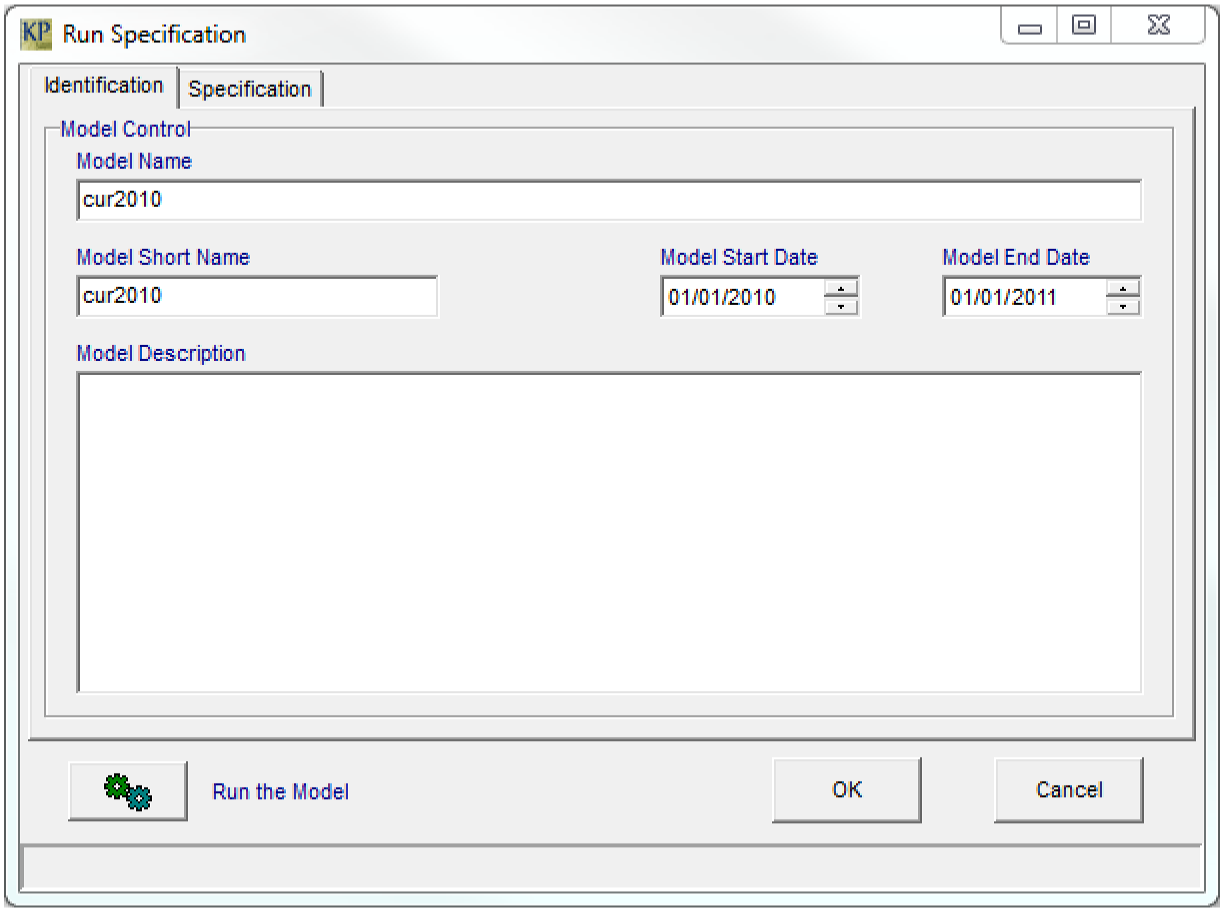
Identification
On the Identification tab of the Run Specification screen, set the:

- **Model Name,** listed on the main screen and used to identify the Model Allocation Run. Use abbreviations that describe the model parameters, i.e., “CY01RegShare” instead of “Model02.”

- **Model Short Name,** used to assign output file names by appending the short name of the Allocation Run to the output file names. For example, the file “CY01RegShareMatch.out” contains the match run results for an allocation run with Model Short Name "CY01RegShare."

- **Model Start Date,** the beginning date of this Model Allocation Run.

- **Model End Date,** the ending date of this Model Allocation Run.



Model Name: cur2010
Model Short Name: cur2010
Model Start Date: 01/01/2010
Model End Date: 01/01/2011
Model Description:

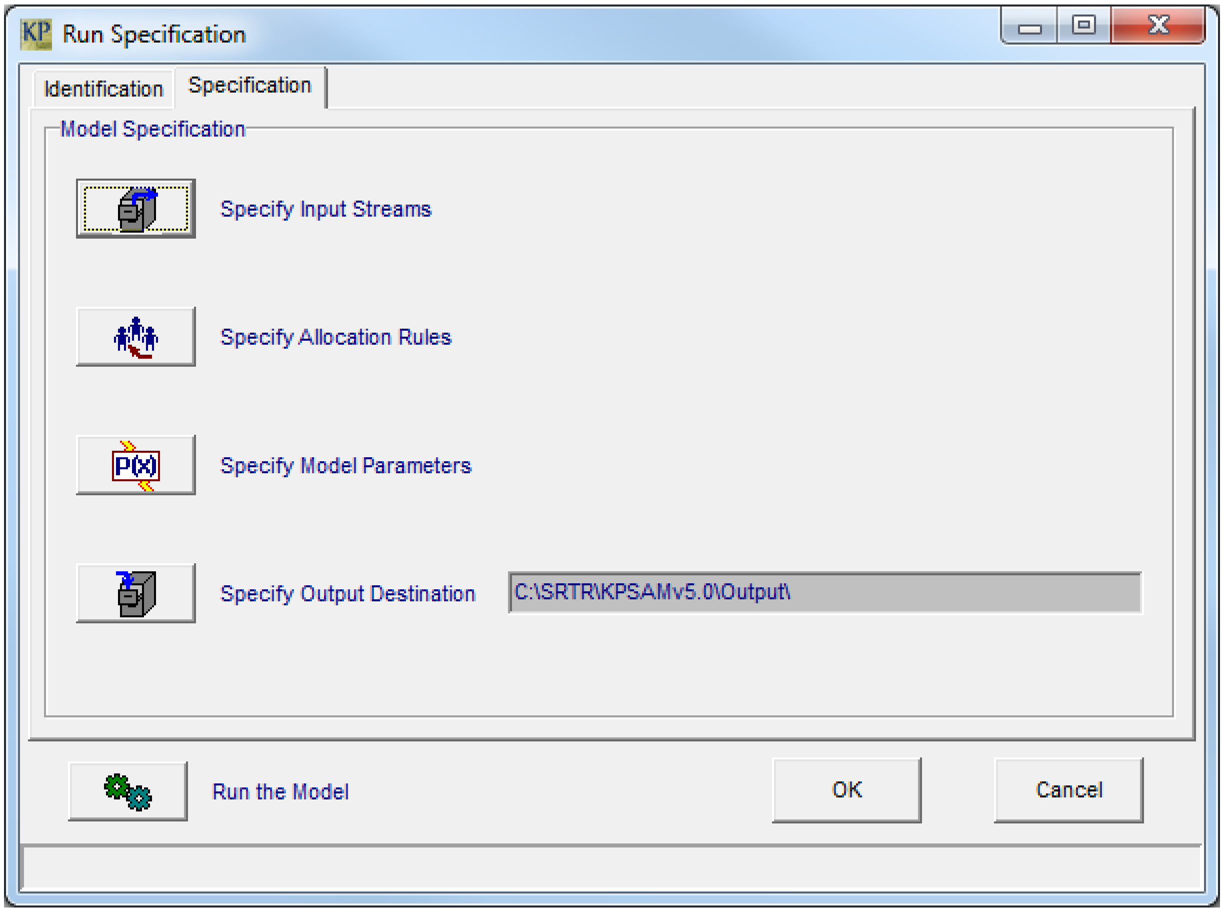
If any Allocation Run packages have been defined previously, the program will list them on this screen. In that case, you also may proceed by selecting one of the existing Allocation Run packages and then selecting the Open Box button (immediately below the Package button). This will take you to the Allocation Run Specification screen, where you may run that package or edit its definition.

The remaining buttons allow you to duplicate an Allocation Run package, delete an Allocation Run package, run a Queue and create a Summary file.

Select the **Specification** tab to continue.

Specification

The next few pages describe setting model specification parameters.



C:\SRTR\KPSAMv5.0\Output\

Select the **Specify Input Streams** button to continue.

1.6.2 Input Selection

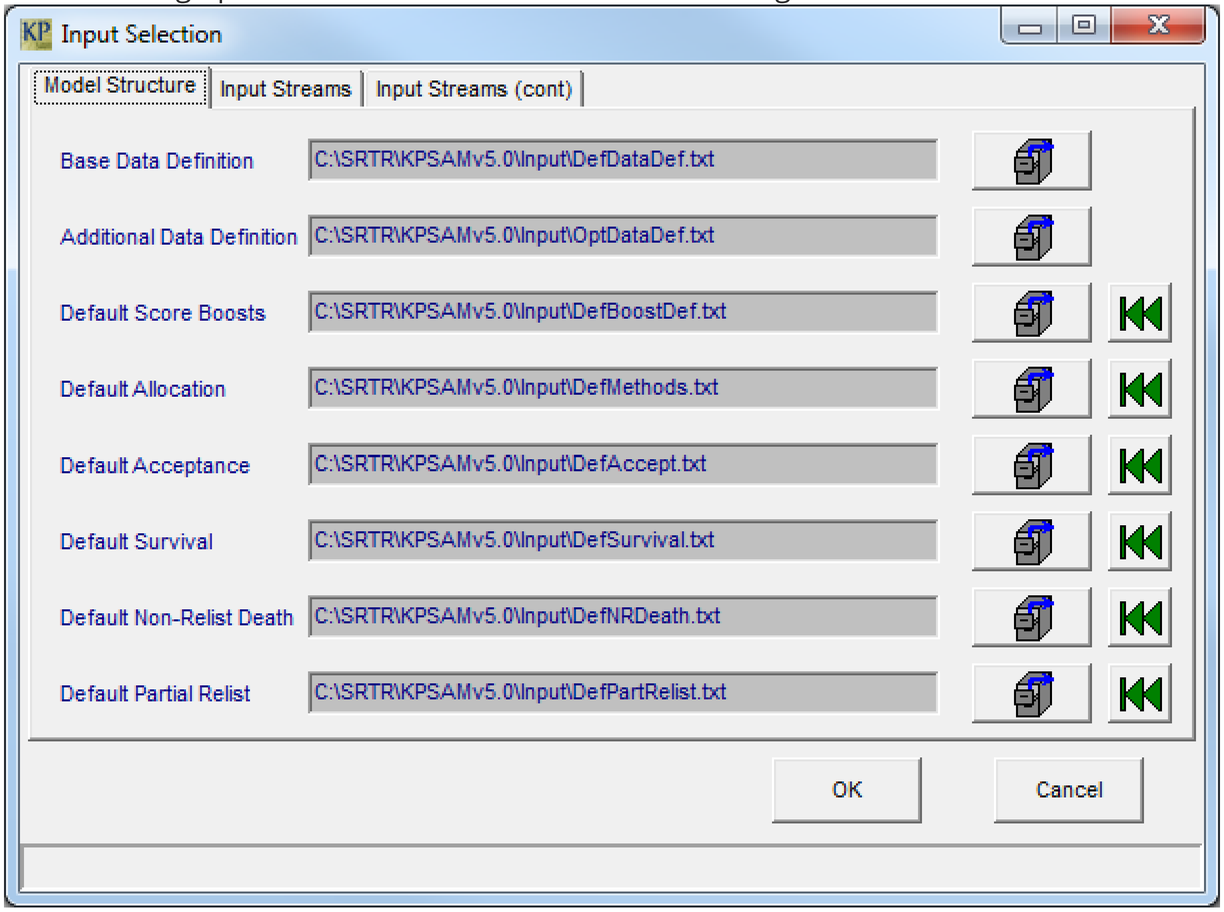
Run Specification: Input Selection

The Input Selection screen contains two tabs, which are used to select the files from which the program will read its input data for this Allocation Run.

Model Structure

The Model Structure tab allows you to specify where the Base and Additional Data Definition, Score Boost, Allocation, Acceptance, and Survival files are located. See Chapter 6, Input File Specifications, for more detailed information about each input file.

To change a data source from its default value, select the filing cabinet button on the right. This will bring up a standard Windows file selection dialog.



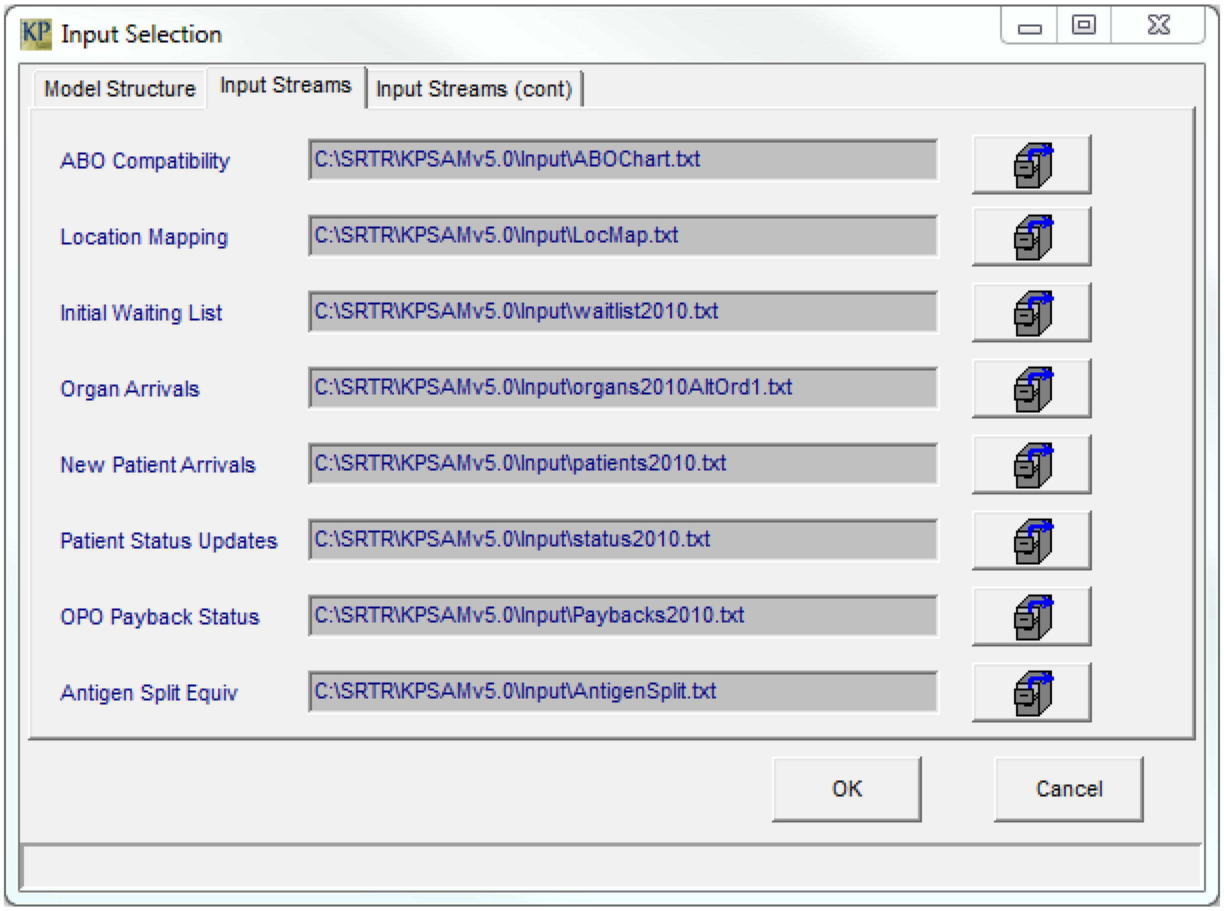
Base Data Definition	C:\SRTR\KPSAMv5.0\Input\DefDataDef.txt
Additional Data Definition	C:\SRTR\KPSAMv5.0\Input\OptDataDef.txt
Default Score Boosts	C:\SRTR\KPSAMv5.0\Input\DefBoostDef.txt
Default Allocation	C:\SRTR\KPSAMv5.0\Input\DefMethods.txt
Default Acceptance	C:\SRTR\KPSAMv5.0\Input\DefAccept.txt
Default Survival	C:\SRTR\KPSAMv5.0\Input\DefSurvival.txt
Default Non-Relist Death	C:\SRTR\KPSAMv5.0\Input\DefNRDeath.txt
Default Partial Relist	C:\SRTR\KPSAMv5.0\Input\DefPartRelist.txt

The double green arrows on the right will reload information from these files. You might use this, for instance, if you have made changes to the post-graft survival calculations (discussed on page 27), and then decide to start over with the default survival calculations.

Next, select the **Input Streams** tab.

Input Streams

The Input Streams tab allows you to specify where the ABO Compatibility, Location Mapping, Initial Waiting List, Organ Arrivals, New Patient Arrivals, Patient Status Updates, Initial Payback Tally, and Antigen Split Equivalencies files are located. To change a location, click on the filing cabinet icon to the right of the file name. See Chapter 6, Input File Specifications, for more detailed information about each input file.


Input Selection

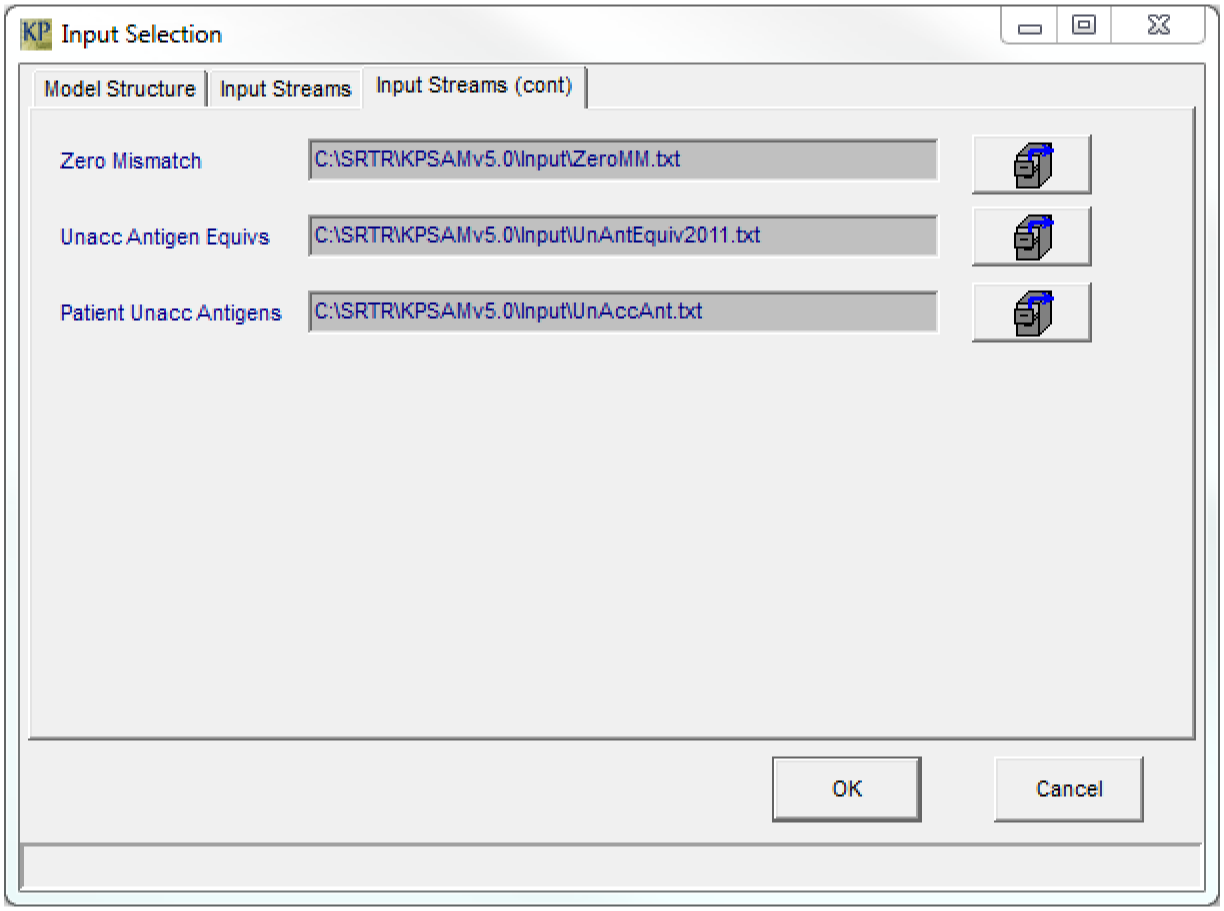
Model Structure | Input Streams | Input Streams (cont)

ABO Compatibility	C:\SRTR\KPSAMv5.0\Input\ABOChart.txt
Location Mapping	C:\SRTR\KPSAMv5.0\Input\LocMap.txt
Initial Waiting List	C:\SRTR\KPSAMv5.0\Input\waitlist2010.txt
Organ Arrivals	C:\SRTR\KPSAMv5.0\Input\organs2010AltOrd1.txt
New Patient Arrivals	C:\SRTR\KPSAMv5.0\Input\patients2010.txt
Patient Status Updates	C:\SRTR\KPSAMv5.0\Input\status2010.txt
OPO Payback Status	C:\SRTR\KPSAMv5.0\Input\Paybacks2010.txt
Antigen Split Equiv	C:\SRTR\KPSAMv5.0\Input\AntigenSplit.txt

Multiple sample organ arrivals files have been provided with this installation. The same organs become available in each of the files, but their arrivals dates and times have been shuffled in each of the files in order to more closely simulate the randomness with which donor organs become available. The user may choose to specify that KPSAM use a different ordering of organ arrivals per iteration by choosing the organ file with the phrase “AltOrd1” in its name as the organ arrivals file on this screen. KPSAM will then use AltOrd1 for its first iteration, AltOrd2 for its second iteration, etc.

Input Streams (cont.)

The third tab allows you to specify where the zero-mismatch, unacceptable antigen equivalences, and patients’ unacceptable antigens updates files are located. To change the location, click on the filing cabinet icon to the right of the file name. See Chapter 6, Input File Specifications, of this document for more detailed information about each input file.



Zero Mismatch: C:\SRTR\KPSAMv5.0\Input\ZeroMM.txt

Unacc Antigen Equivs: C:\SRTR\KPSAMv5.0\Input\UnAntEquiv2011.txt

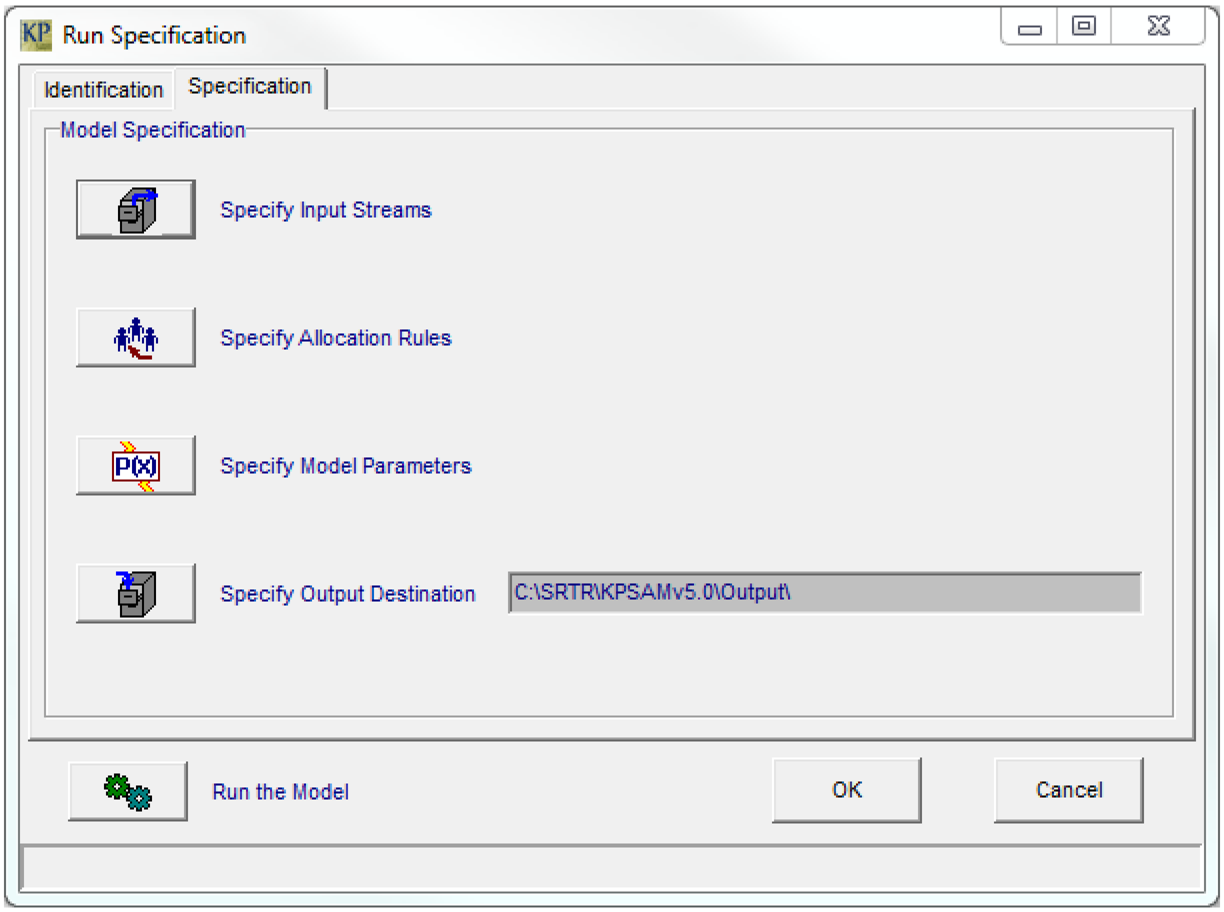
Patient Unacc Antigens: C:\SRTR\KPSAMv5.0\Input\UnAccAnt.txt

1.6.3 Allocation Rules

<u>**Run Specification: Allocation Rules**</u>

KPSAM provides interfaces for adjusting allocation rules and other model parameters. All of these settings are also specified in the default input files, as discussed in Chapter 6, Input File Specifications. The graphical interfaces may be convenient for small adjustments, but using a text editor to manually copy and edit the input files is recommended for large-scale changes.

To explore the first graphical parameter interface, select the **Specify Allocation Rules** button.



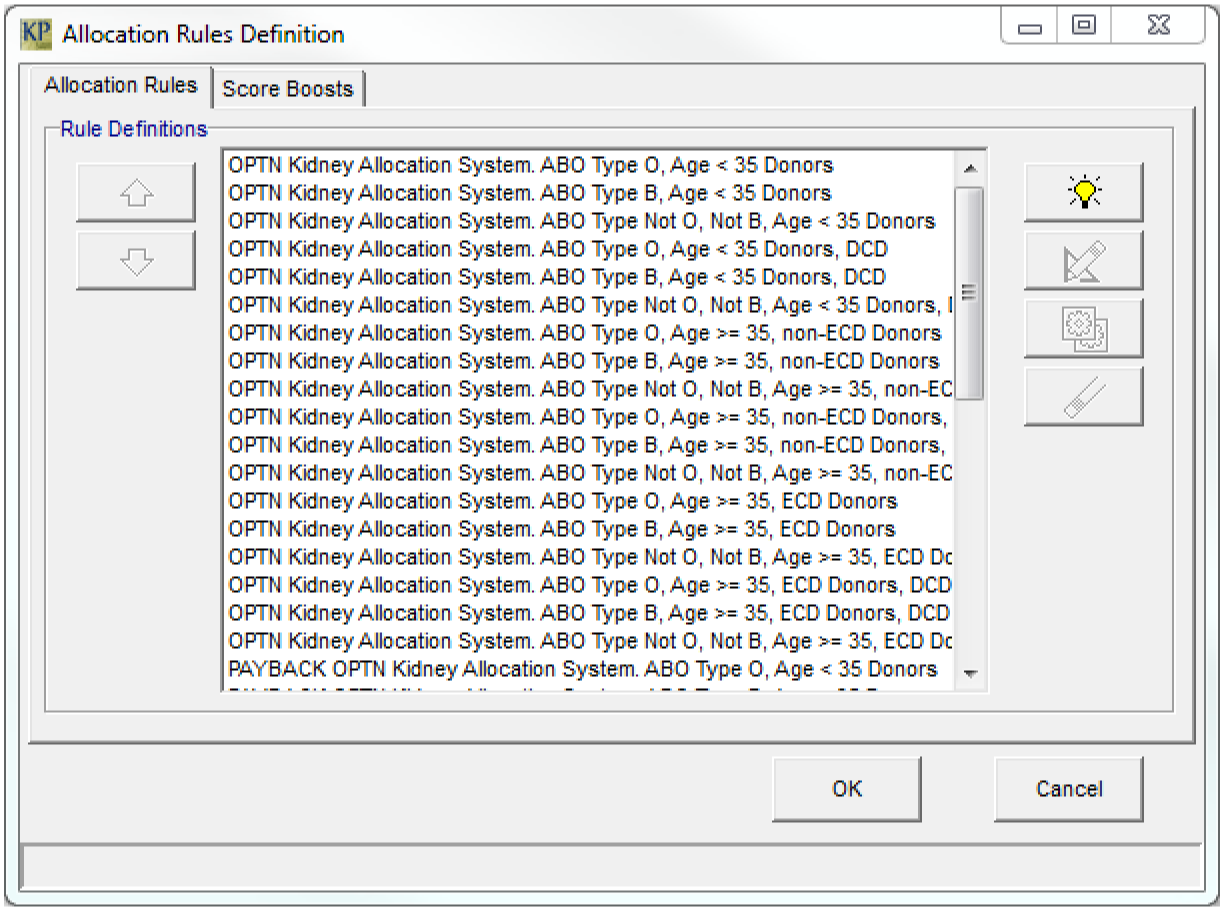
C:\SRTR\KPSAMv5.0\Output\

Allocation Rules Definition

The Allocation Rules dialog contains two tabs. Use the Allocation Rules and Score Boosts tabs to design the rules for the allocation of kidneys and pacreata. This interface corresponds to the input file DefMethods.txt.

Allocation Rules

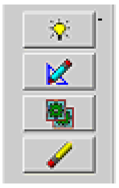
The Allocation rules tab allows you to specify the rules to use. See Chapter 3 for more detailed information about each rule.



OPTN Kidney Allocation System. ABO Type O, Age < 35 Donors
OPTN Kidney Allocation System. ABO Type B, Age < 35 Donors
OPTN Kidney Allocation System. ABO Type Not O, Not B, Age < 35 Donors
OPTN Kidney Allocation System. ABO Type O, Age < 35 Donors, DCD
OPTN Kidney Allocation System. ABO Type B, Age < 35 Donors, DCD
OPTN Kidney Allocation System. ABO Type Not O, Not B, Age < 35 Donors, I
OPTN Kidney Allocation System. ABO Type O, Age >= 35, non-ECD Donors
OPTN Kidney Allocation System. ABO Type B, Age >= 35, non-ECD Donors
OPTN Kidney Allocation System. ABO Type Not O, Not B, Age >= 35, non-EC
OPTN Kidney Allocation System. ABO Type O, Age >= 35, non-ECD Donors,
OPTN Kidney Allocation System. ABO Type B, Age >= 35, non-ECD Donors,
OPTN Kidney Allocation System. ABO Type Not O, Not B, Age >= 35, non-EC
OPTN Kidney Allocation System. ABO Type O, Age >= 35, ECD Donors
OPTN Kidney Allocation System. ABO Type B, Age >= 35, ECD Donors
OPTN Kidney Allocation System. ABO Type Not O, Not B, Age >= 35, ECD Dc
OPTN Kidney Allocation System. ABO Type O, Age >= 35, ECD Donors, DCD
OPTN Kidney Allocation System. ABO Type B, Age >= 35, ECD Donors, DCD
OPTN Kidney Allocation System. ABO Type Not O, Not B, Age >= 35, ECD Dc
PAYBACK OPTN Kidney Allocation System. ABO Type O, Age < 35 Donors

To explore the first graphical parameter interface, select the **Specify Allocation Rules** button.

The buttons on this screen are defined as follows:



💡	Create new
✎	Edit
⧉	Copy
✏	Delete

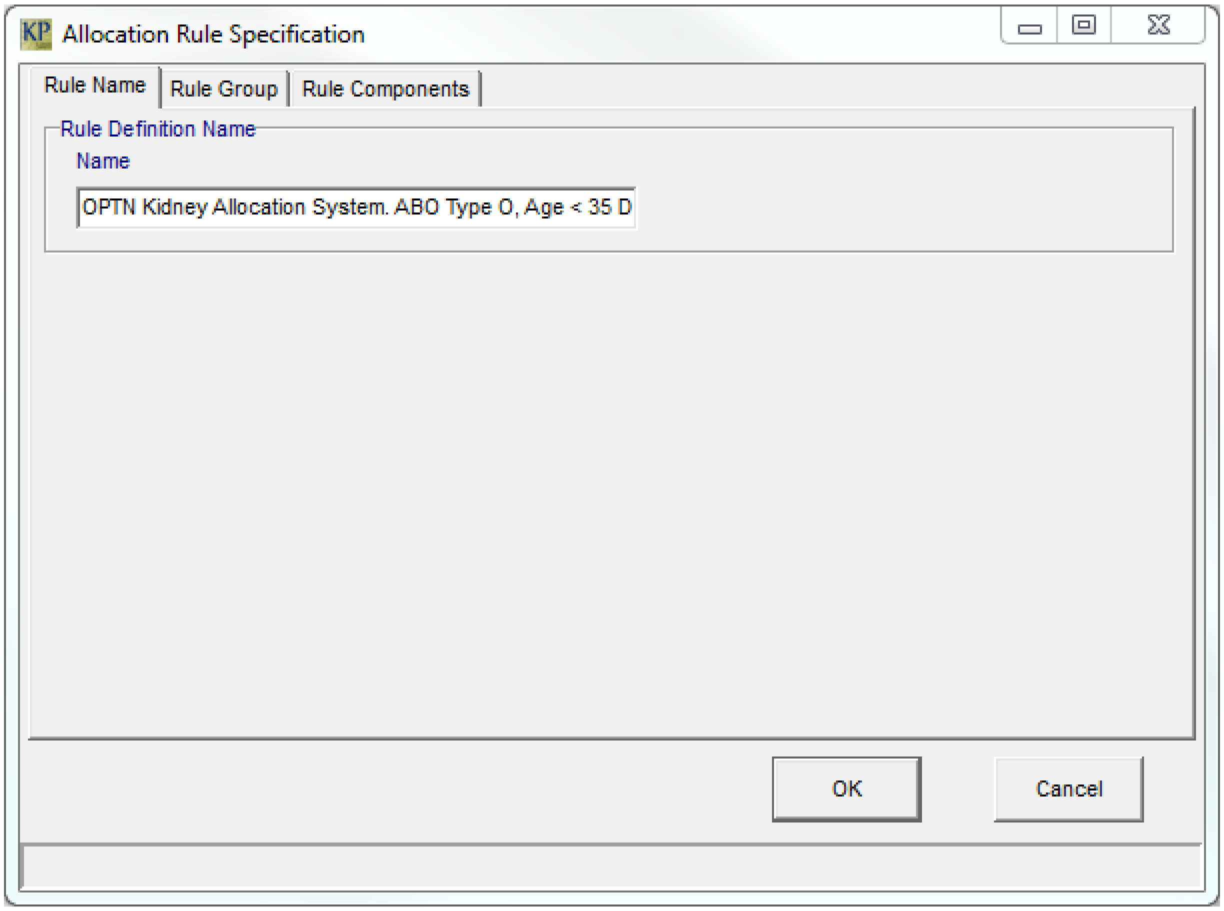
Highlight one of the Allocation Rules, such as “OPTN Kidney Allocation System. ABO Type O, Age < 35 Donors” and select the edit button (blue pencil) to continue.

Allocation Rule Specification

The Allocation Rule Specification screen contains three tabs that are used to define the rules for organ allocation.

Rule Name

Use the Rule Name tab to specify the name of this allocation rule. The name should describe the set of donors to whom this rule applies.



OPTN Kidney Allocation System. ABO Type O, Age < 35 D

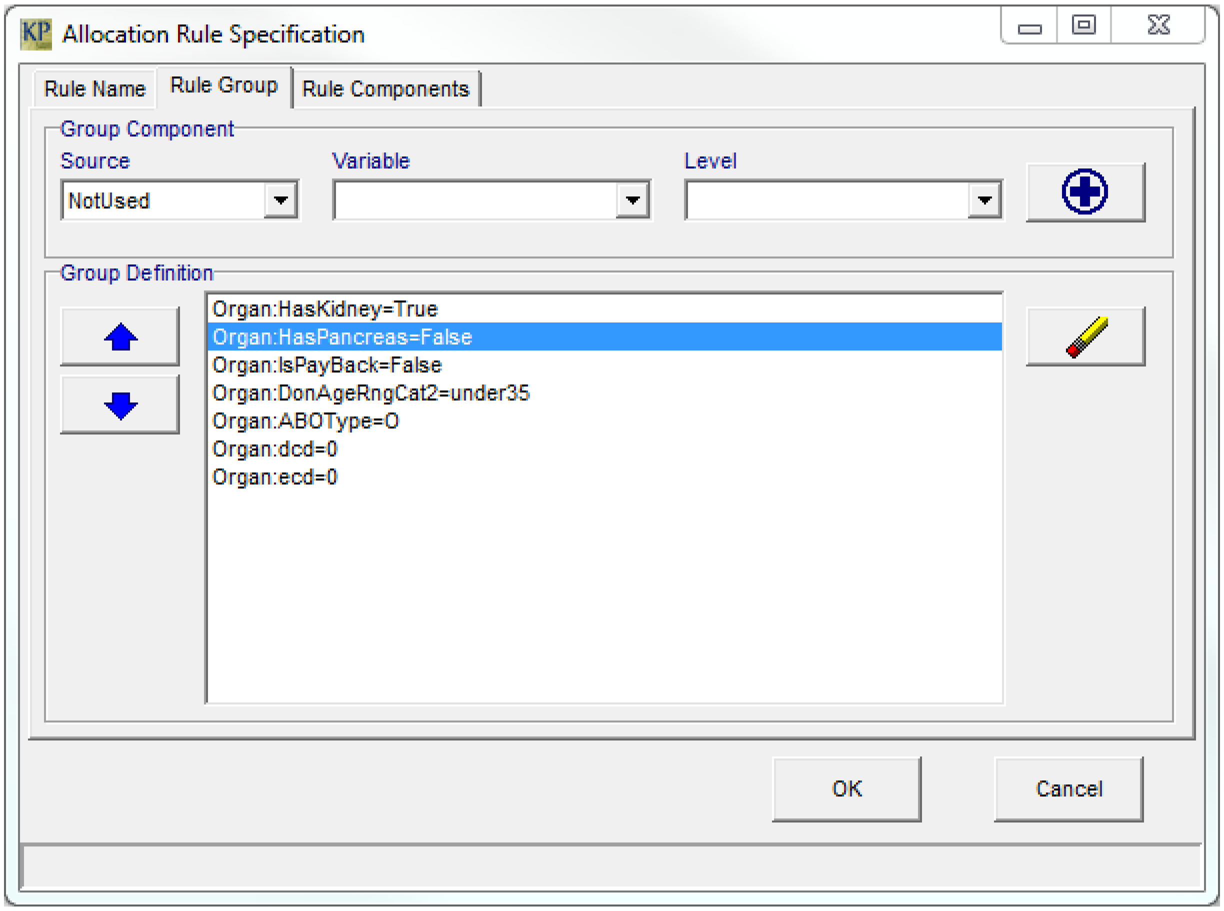
Next, select the **Rule Group** tab.

Rule Group

Use the Rule Group tab to define the group of donors to whom this allocation rule applies.

The source must be the organ record. Choose a variable from the drop-down list. This list contains all fields whose usage has been defined as “category” in the default or optional data definitions files. Choose a level for this variable from the level drop-down list. These, too, have been defined in the default or optional data definitions files. See Chapter 6, Input File Specifications, for more detailed information about the data definitions files.

You may combine any number of rules using the plus sign button at the right to narrow the donor list to which this rule applies. An organ donor must meet all of the group definitions for this allocation rule to apply.



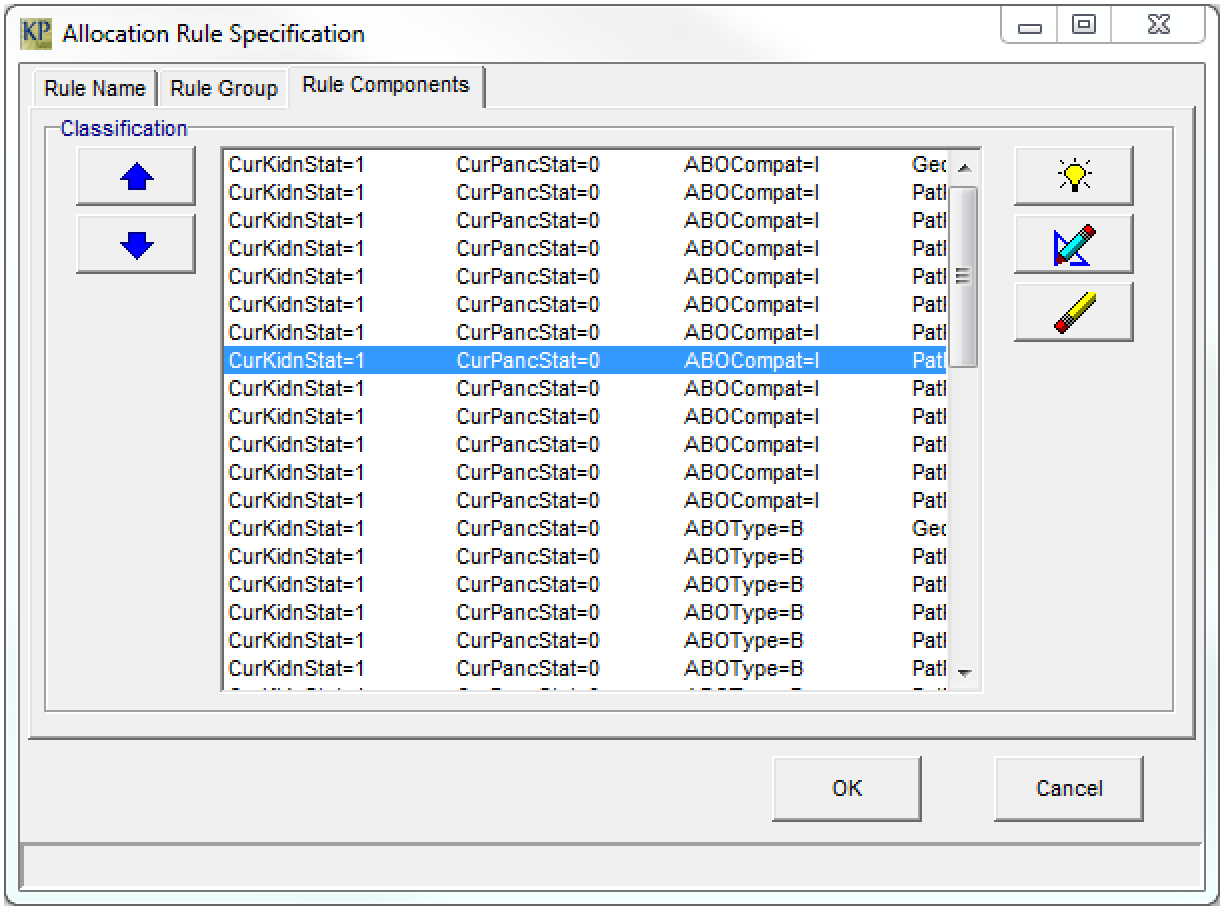
Source: NotUsed

Organ:HasKidney=True
Organ:HasPancreas=False
Organ:IsPayBack=False
Organ:DonAgeRngCat2=under35
Organ:ABOType=O
Organ:dcd=0
Organ:ecd=0

Next, select the **Rule Components** tab.

Rule Components

The Rule Components tab displays the ordered list of patient categories for the selected allocation rule. To change the ordering, select a category and use the up- and down-arrow keys on the left to promote or demote the category. To create a new category, select the create button (light bulb) on the right side. To edit a category, select the category and then select the edit button (blue pencil) on the right side. To delete a category, select it and then select the erase button (pencil eraser). See Chapter 3 for more detailed information about the allocation methods.



CurKidnStat=1	CurPancStat=0	ABOCompat=I	Ged
CurKidnStat=1	CurPancStat=0	ABOCompat=I	Pati
CurKidnStat=1	CurPancStat=0	ABOCompat=I	Pati
CurKidnStat=1	CurPancStat=0	ABOCompat=I	Pati
CurKidnStat=1	CurPancStat=0	ABOCompat=I	Pati
CurKidnStat=1	CurPancStat=0	ABOCompat=I	Pati
CurKidnStat=1	CurPancStat=0	ABOCompat=I	Pati
CurKidnStat=1	CurPancStat=0	ABOCompat=I	Pati
CurKidnStat=1	CurPancStat=0	ABOCompat=I	Pati
CurKidnStat=1	CurPancStat=0	ABOCompat=I	Pati
CurKidnStat=1	CurPancStat=0	ABOCompat=I	Pati
CurKidnStat=1	CurPancStat=0	ABOCompat=I	Pati
CurKidnStat=1	CurPancStat=0	ABOCompat=I	Pati
CurKidnStat=1	CurPancStat=0	ABOType=B	Ged
CurKidnStat=1	CurPancStat=0	ABOType=B	Pati
CurKidnStat=1	CurPancStat=0	ABOType=B	Pati
CurKidnStat=1	CurPancStat=0	ABOType=B	Pati
CurKidnStat=1	CurPancStat=0	ABOType=B	Pati
CurKidnStat=1	CurPancStat=0	ABOType=B	Pati

Highlight one of the rule components, and then select the edit button to continue.

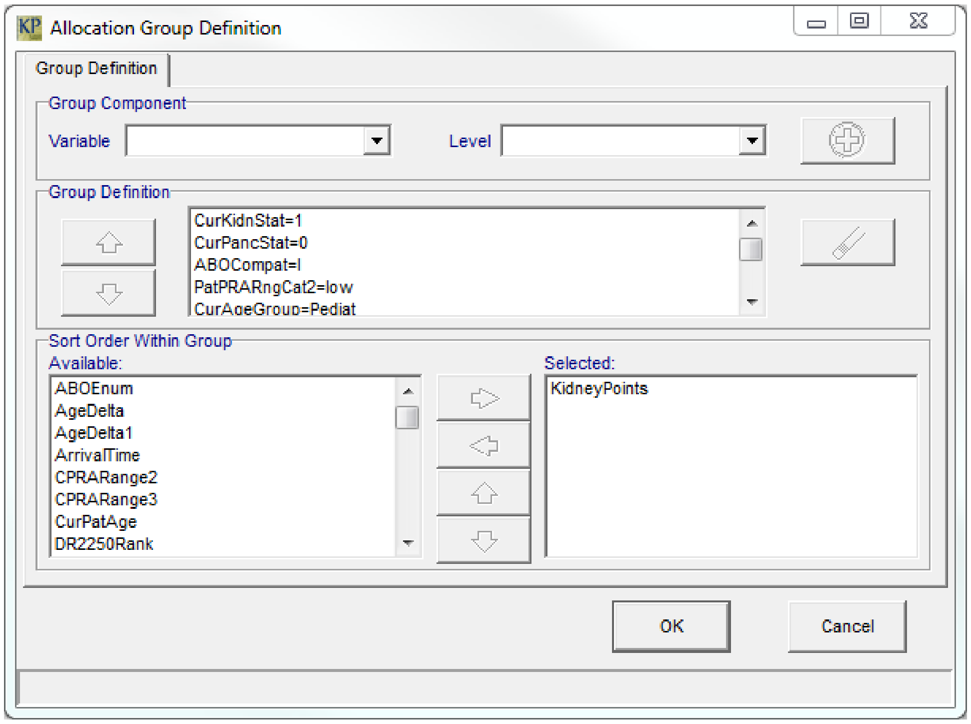
<u>Allocation Group Definition</u>

Group Definition

The Group Definition tab allows users to define the patient categories. The drop-down list of variables contains all patient variables whose usage has been defined as "category" in the default or optional data definition files. Choose a variable from the list, and then choose a level for this variable from the level drop-down list. Rearranging the group definitions list using the up- and down-arrows to the left changes the order in which the criteria are checked. Ordering the more restrictive criteria closer to the top of the list may speed processing. These variables have been defined in the default or optional data definitions files. See Chapter 6, Input File Specifications, for more detailed information about the data definitions files.

Within each patient subgroup, you may want to order the patients by different criteria. Use the Sort Order Within Group section of the screen for subgroup ordering. To add a patient characteristic to the list, highlight that characteristic in the available list, then select the right-arrow button. To remove a characteristic, highlight it in the selected list, then select the left-arrow button. To promote or demote a characteristic, highlight it in the selected list, then select the up- or down-arrow key.

The next section describes how to boost the value of any of the subgroup fields for patient and organ combinations with certain characteristics.



CurKidnStat=1
CurPancStat=0
ABOCompat=I
PatPRARngCat2=low
CurAgeGroup=Pediat

ABOEnum
AgeDelta
AgeDelta1
ArrivalTime
CPRARange2
CPRARange3
CurPatAge
DR2250Rank

KidneyPoints

From the Allocation Group Definition screen, select the OK or Cancel button to return to the Allocation Rule Specification screen. Select the OK or Cancel button from there to re-turn to the Allocation Rules Definition screen.

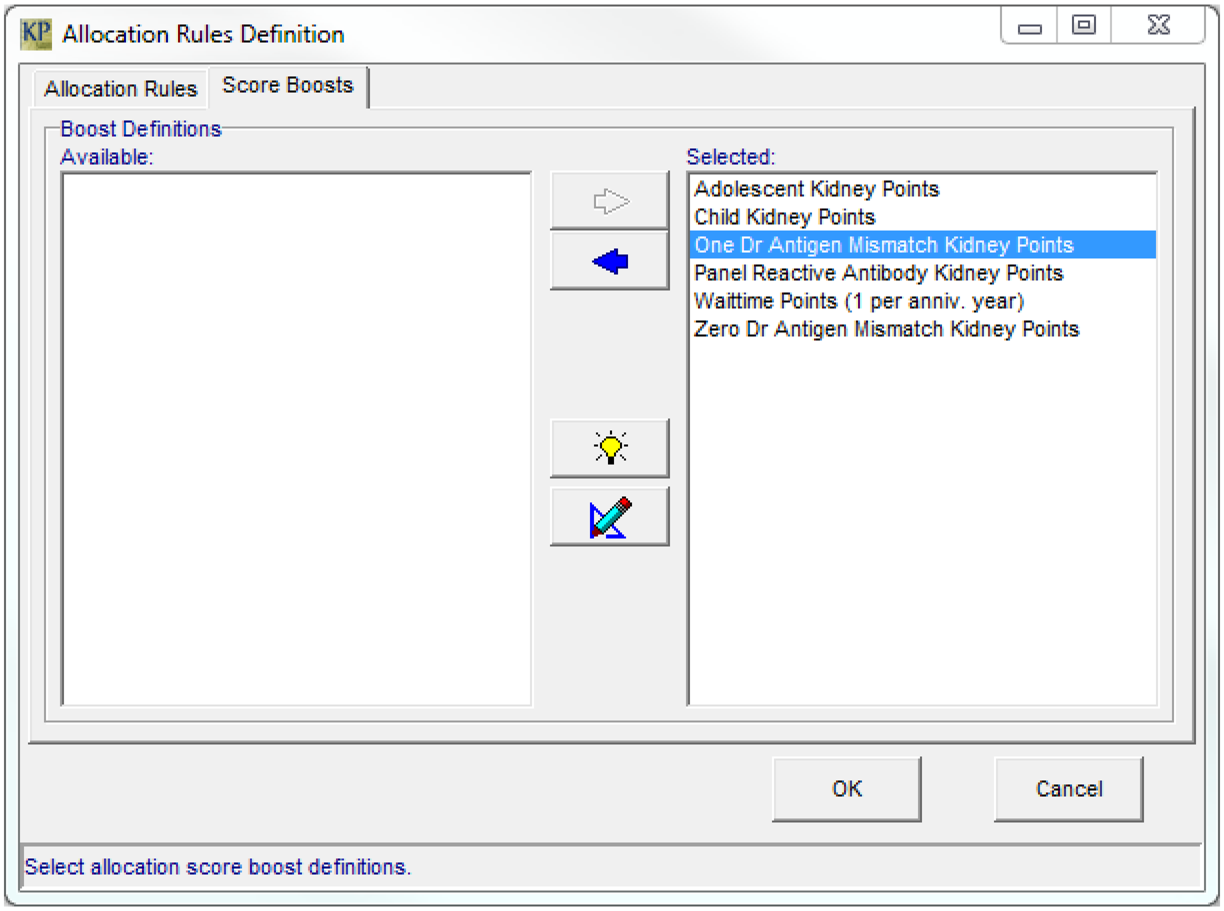
Next, select the **Score Boosts** tab from the Allocation Rules Definition screen.

Allocation Rules Definition

Score Boosts

Use the Score Boosts tab to increase the value of certain patient fields. Those fields may then be used for subgroup sorting in the allocation process. Refer to the previous section, Allocation Group Definition, for information on how to use those fields in the allocation process.

Here you will define the patient and organ characteristics that must be present, and the calculation for the points those patients will gain. This section of the interface corresponds to the DefBoostDef.txt file.



Adolescent Kidney Points
Child Kidney Points
One Dr Antigen Mismatch Kidney Points
Panel Reactive Antibody Kidney Points
Waittime Points (1 per anniv. year)
Zero Dr Antigen Mismatch Kidney Points

Select allocation score boost definitions.

Highlight a score boost from either the available list or the selected list and select the edit button (blue pencil).

The Panel Reactive Antibody Kidney Points in the screen capture above is an example of a score boost. The linear equation score boosts cannot be entered, viewed, or modified using the KPSAM input panels. They must be entered using the DefBoostDef.txt input file, instead. Please refer to the Allocation Score Boost Definition section of the Input File Specification chapter for more information.

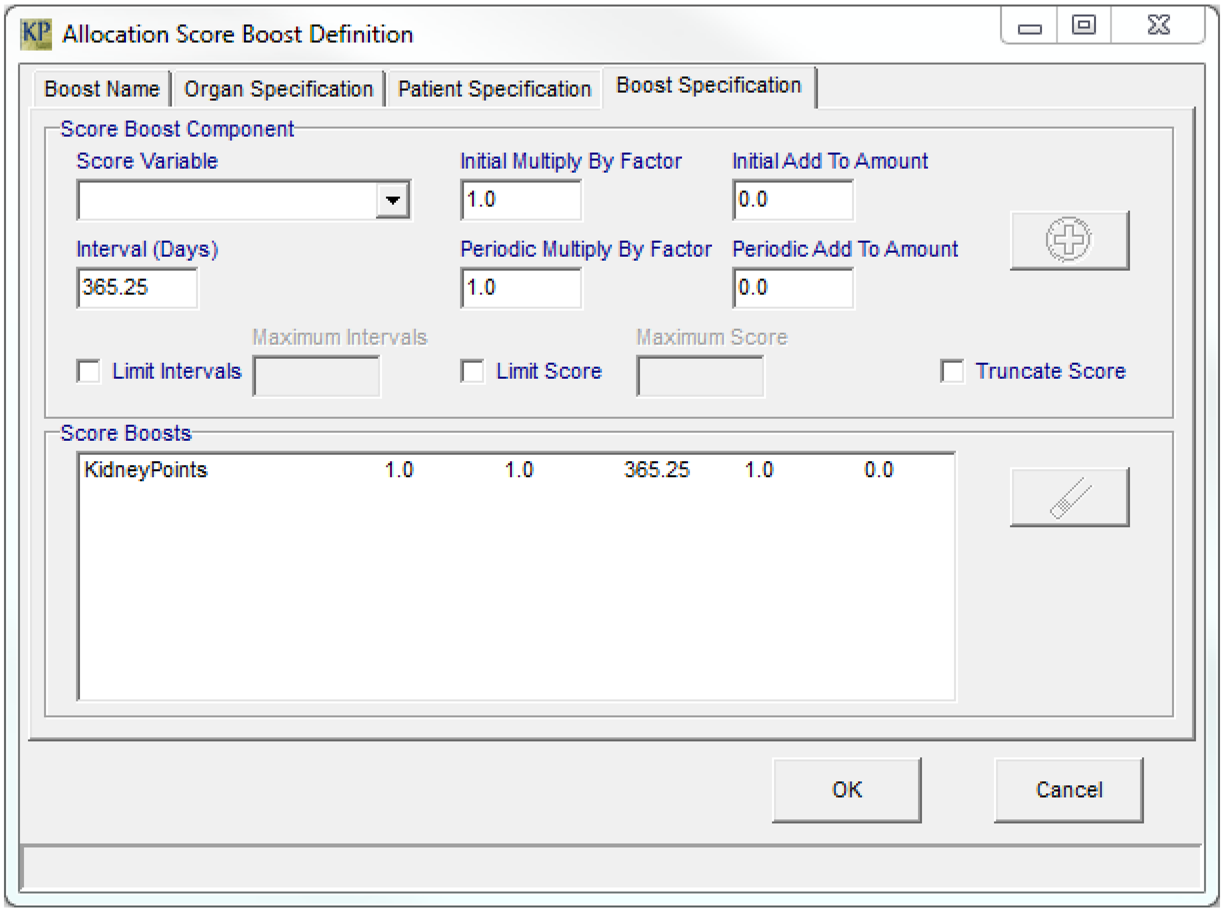
The Boost Name, Organ Specification and Patient Specification tabs on the next screen are similar to the Rule Name and Rule Group tabs in the Allocation Rule Specification. Please refer to the Allocation Rule Specification, earlier in this chapter, for more information on those two tabs.

Select the **Boost Specification** tab to continue.

Allocation Score Boost Definition

Boost Specification

The Boost Specification tab is used to specify which patient characteristic(s) will be increased, and by how much. You may multiply the field by a factor and/or add an amount to the field. In addition, you can specify that the field increase at a given interval if the patients remain on the waiting list. You may also set upper limits on the number of intervals or on the score itself, and you may truncate a calculated score.



Initial Multiply By Factor: 1.0
Initial Add To Amount: 0.0
Interval (Days): 365.25
Periodic Multiply By Factor: 1.0
Periodic Add To Amount: 0.0

KidneyPoints	1.0	1.0	365.25	1.0	0.0

We have completed our allocation rules definitions. Next, we will explore the model parameters you may specify for random number generation, acceptance probability, and post-graft survival.

Select the OK or Cancel button to return to the Allocation Rules Definition screen, then select OK or Cancel again to return to the Run Specification screen. Finally choose the **Specify Model Parameters** button to continue.

1.6.4 Model Parameter Specification

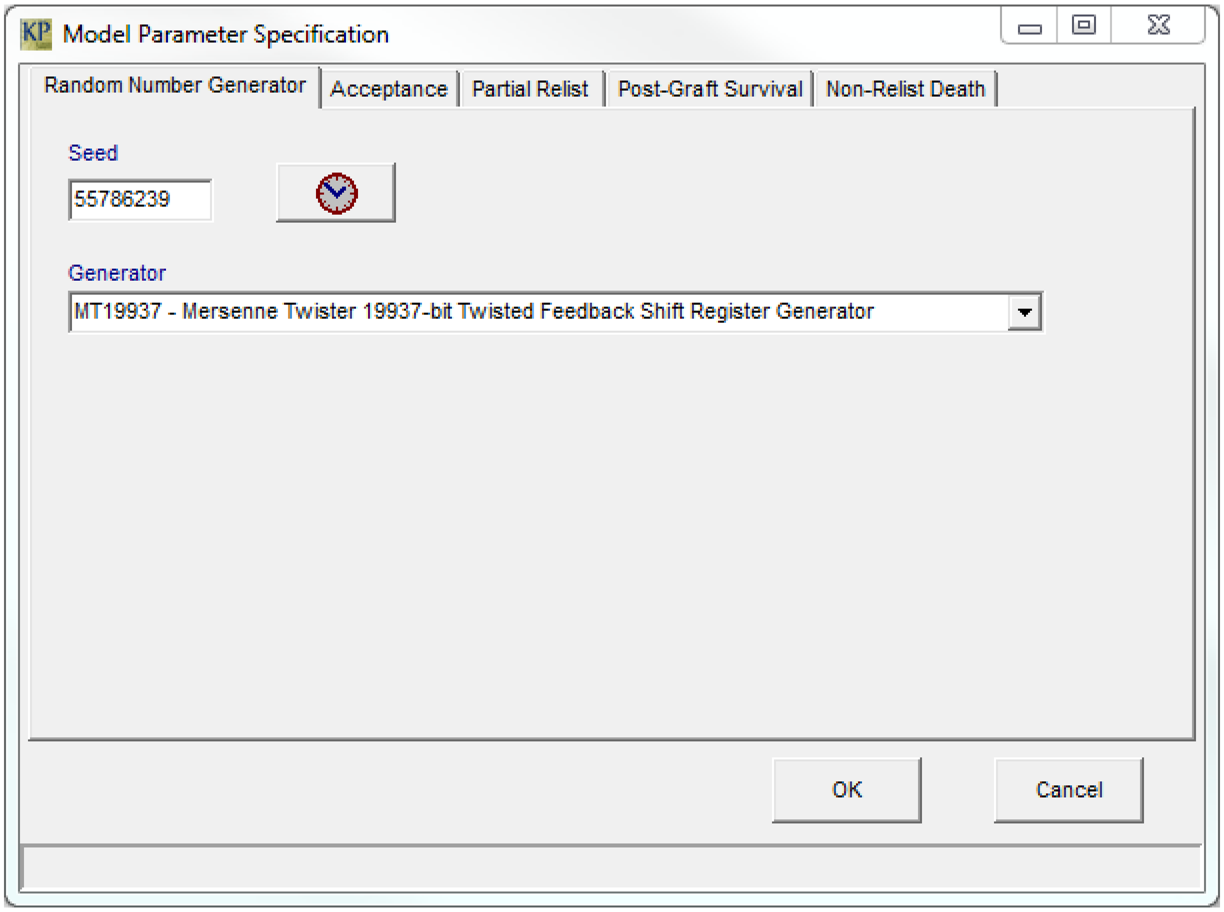
Random Number Generator

This list of choices allows you to choose how the program will generate random numbers for allocation. Random numbers are used in several places in this simulation; for example, if the probability of organ acceptance for a certain organ/recipient pair is calculated as 70%, the program will use random numbers from this generator to decide whether that recipient accepts that particular organ. For most purposes, the choice of generator does not matter.

The default generator, MT19937, is preferred, although the stream of random numbers produced by any of these generators should be adequate.

If you wish to duplicate the results of a run exactly, you must provide the same seed and the same generator.

You can accept the existing seed, enter a new seed in the window, or select the clock button and the program will select a new seed. See Chapter 2.1 for information on how the model uses the random number generator.



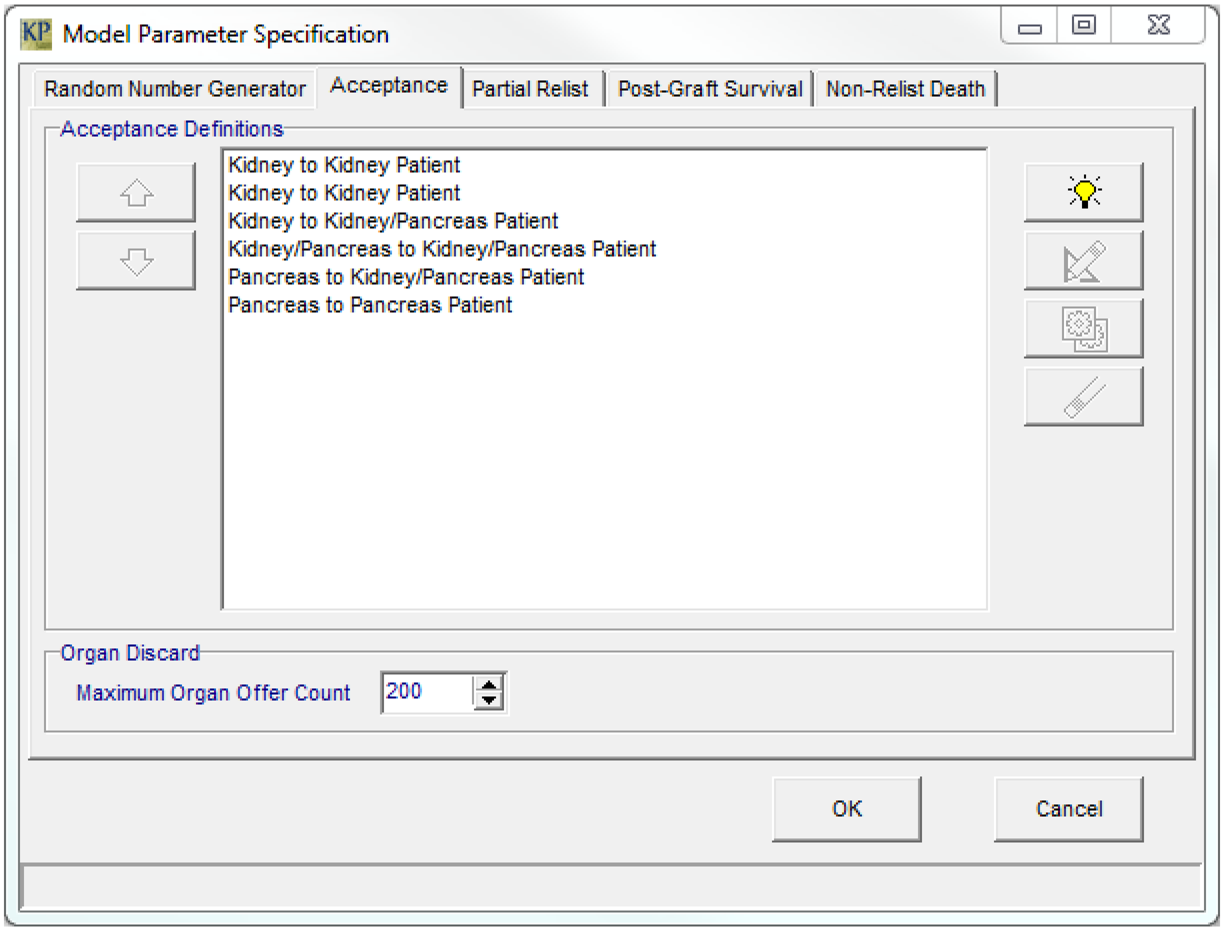
Seed: 55786239

Generator: MT19937 - Mersenne Twister 19937-bit Twisted Feedback Shift Register Generator

Next, select the **Acceptance** tab.

Acceptance

Acceptance probability is determined by model input. The mechanism is described in detail below. You may also limit the number of times a given organ is offered before it is discarded. Lowering this limit will speed processing and result in more discarded organs. The acceptance model you use should reflect the offer limit you set. The default acceptance model parameters are also available in DefAccept.txt.



Kidney to Kidney Patient
Kidney to Kidney Patient
Kidney to Kidney/Pancreas Patient
Kidney/Pancreas to Kidney/Pancreas Patient
Pancreas to Kidney/Pancreas Patient
Pancreas to Pancreas Patient

Organ Discard

Maximum Organ Offer Count: 200

Highlight one of the acceptance models provided and select the edit button.

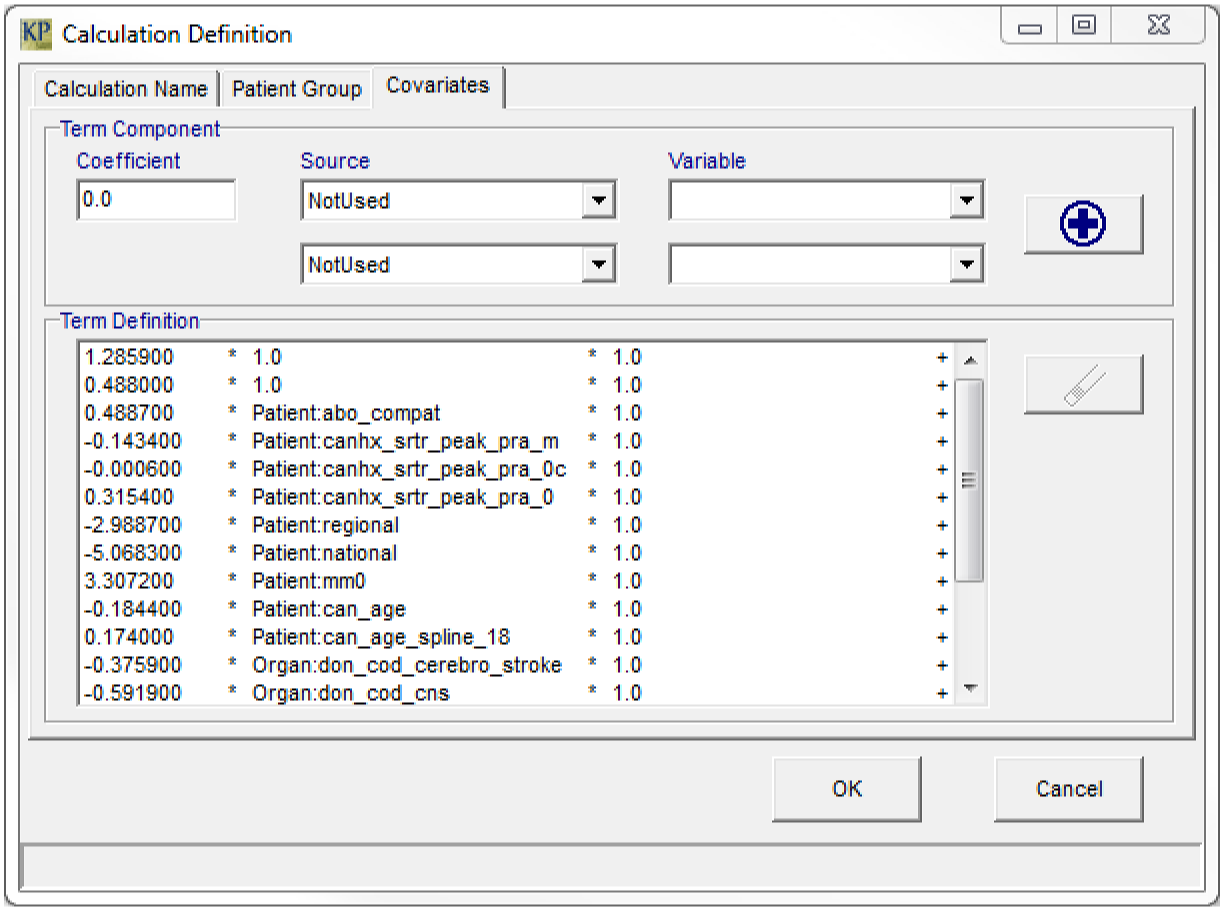
The Calculation Name and Patient Group tabs on the next screen are similar to the Rule Name and Rule Group tabs in the Allocation Rule Specification. Please refer to the Allocation Rule Specification, earlier in this chapter, for more information on those two tabs.

Select the **Covariates** tab to continue.

Calculation Definition (Acceptance)

Covariates

Here you may define an equation to represent organ acceptance probability. Values that may be used in this equation are scalar variables (coefficients), characteristics of the organ or donor, characteristics of the potential organ recipient, and/or values calculated from characteristics of the specific organ and patient combination under consideration. For each organ offered, this equation will be solved by the model, and the resulting value, β, transformed using an inverse logit transformation (exp(β) / 1+exp(β)). That value will then be compared to a random number between 0 and 1. If it is greater than the random number, then the organ is accepted; otherwise, it is refused.



Coefficient: 0.0
Source: NotUsed
Source: NotUsed

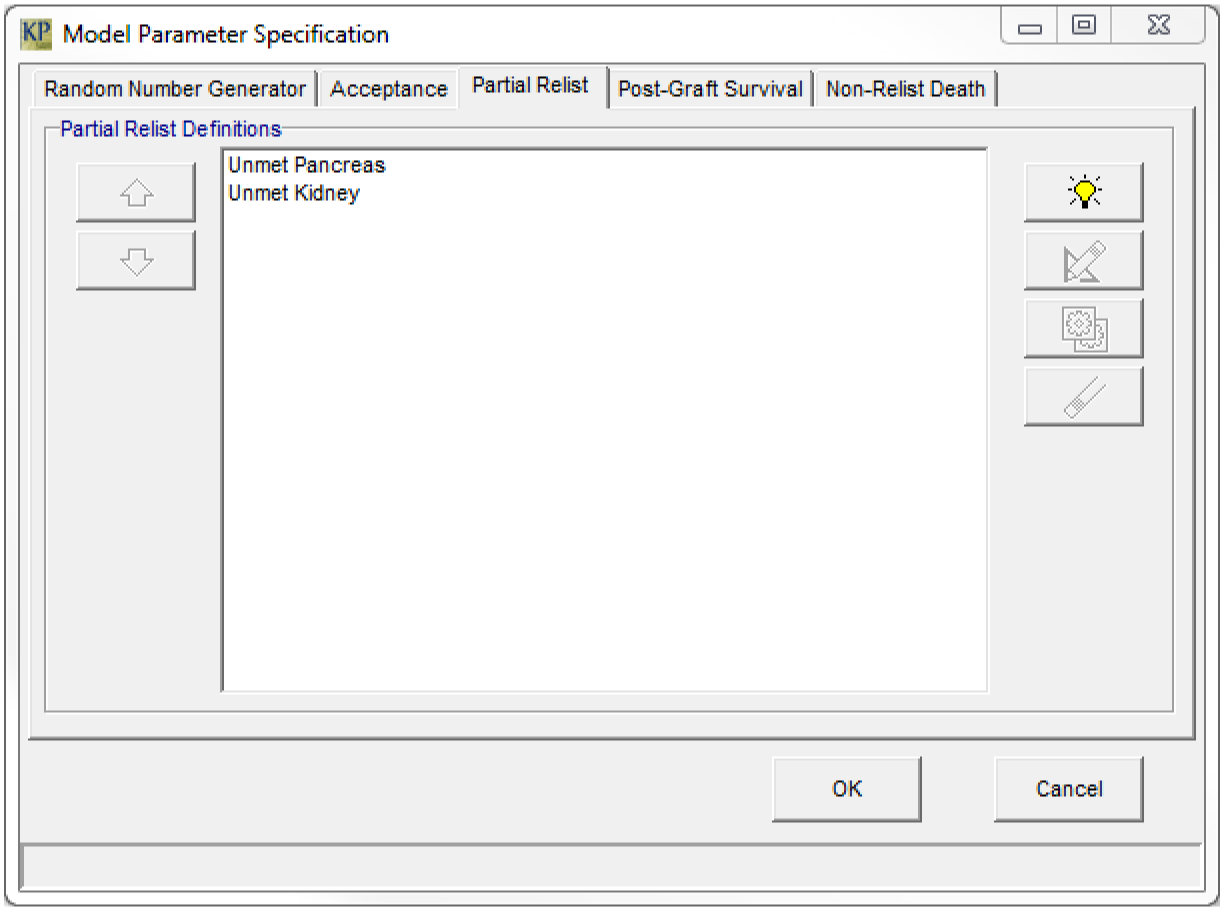
```
1.285900     *  1.0                           *  1.0    +
0.488000     *  1.0                           *  1.0    +
0.488700     *  Patient:abo_compat            *  1.0    +
-0.143400    *  Patient:canhx_srtr_peak_pra_m  *  1.0    +
-0.000600    *  Patient:canhx_srtr_peak_pra_0c *  1.0    +
0.315400     *  Patient:canhx_srtr_peak_pra_0  *  1.0    +
-2.988700    *  Patient:regional              *  1.0    +
-5.068300    *  Patient:national              *  1.0    +
3.307200     *  Patient:mm0                   *  1.0    +
-0.184400    *  Patient:can_age               *  1.0    +
0.174000     *  Patient:can_age_spline_18     *  1.0    +
-0.375900    *  Organ:don_cod_cerebro_stroke  *  1.0    +
-0.591900    *  Organ:don_cod_cns             *  1.0    +
```

Select the OK or Cancel button to return to the Model Parameter Specification screen, then choose the **Post-Graft Survival** tab to continue.

Model Parameter Specification

Partial Relist

On the Partial Relist panel, you may define the calculation to perform to determine whether a kidney-pancreas patient who receives an isolated kidney or an isolated pancreas will immediately relist for the organ not received.



Unmet Pancreas
Unmet Kidney

Highlight one of the Partial Relist definitions provided and select the edit button.

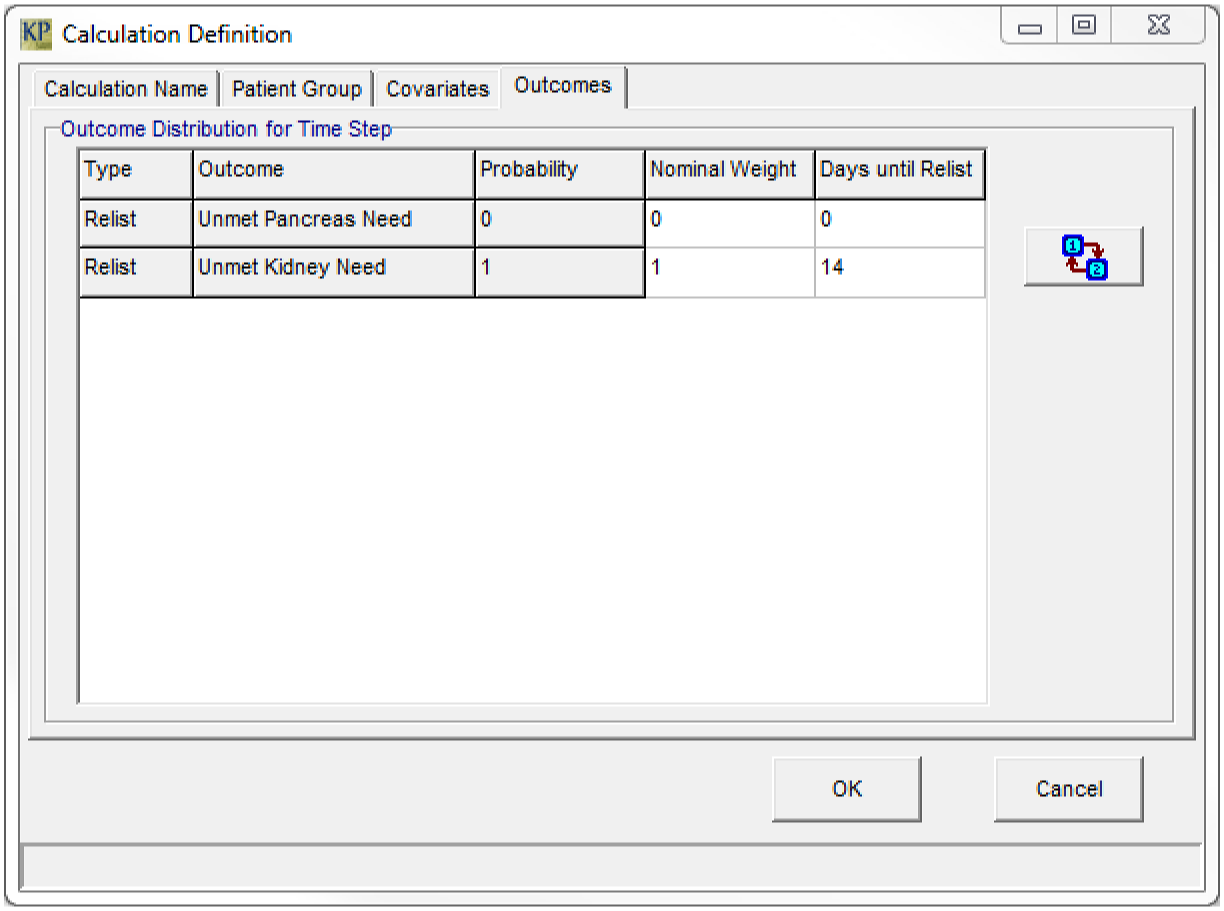
The Calculation Name, Patient Group and Covariates tabs on the next screen are similar to those in the Acceptance Specification. Please refer to the Acceptance Specification, earlier in this chapter, for more information on those tabs.

The Covariates tab, in the case of partial relist, is the definition of the calculation to perform to produce the probability that a KP patient with partially met requirements relists for the organ not received.

Select the **Outcomes** tab to continue.

Calculation Definition (Partial Relist)

Outcomes


Calculation Definition

Outcome Distribution for Time Step

Type	Outcome	Probability	Nominal Weight	Days until Relist
Relist	Unmet Pancreas Need	0	0	0
Relist	Unmet Kidney Need	1	1	14

On the outcomes panel, the probabilities of relisting for unmet need are set to 1 for the organ needed, and 0 for the one, which is not needed. Set the Days until Relist to be the number of days after the initial graft that the patient will relist for the organ not received.

On the following panel, you may specify a series of status updates for the relisted patients.

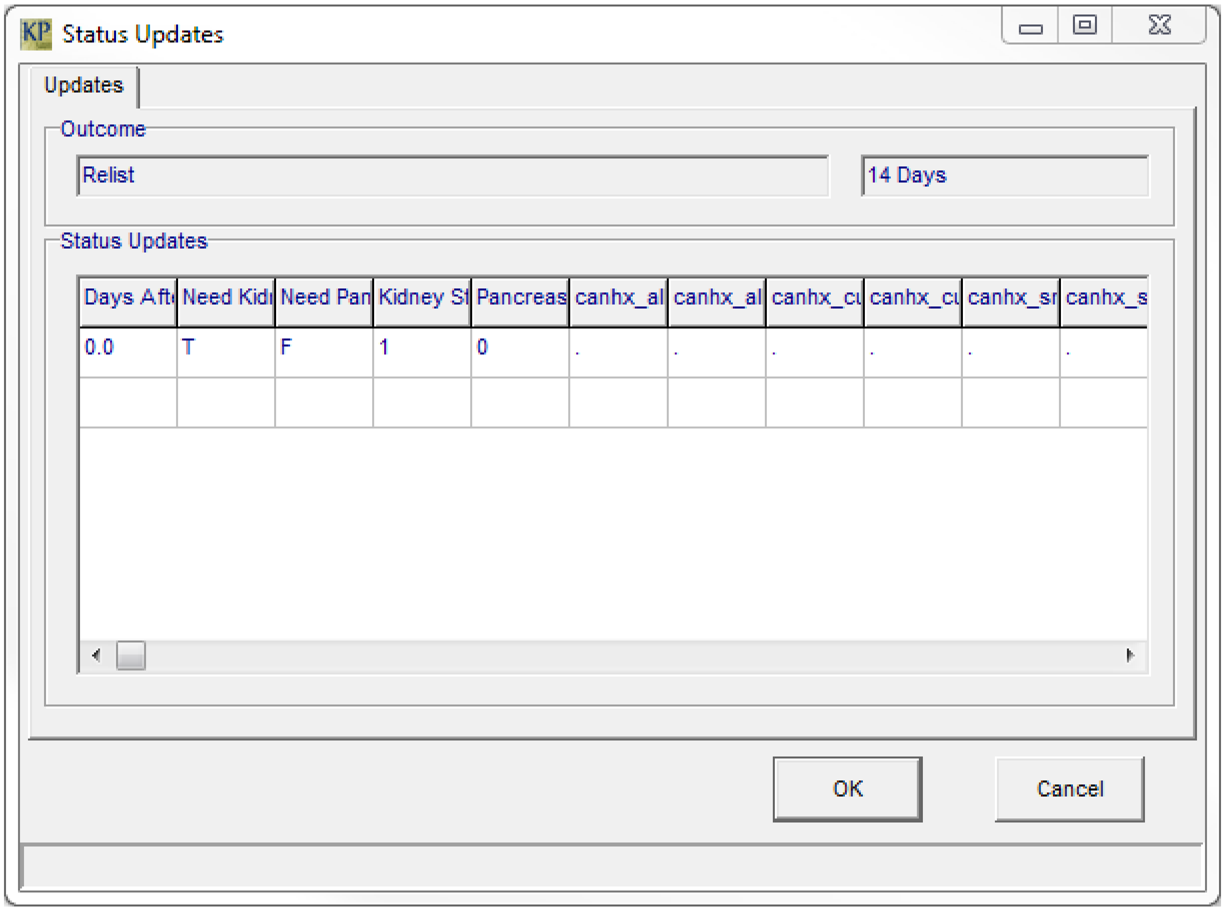
Highlight the Nominal Weight field that is set to 1, and select the **Update** button.

Update

Status Updates

Updates

All status updates that you would like for this patient to receive subsequent to transplant should be defined on this screen. Each update is relative to the date on which the patient is scheduled to relist. Possible status updates are changes from Active to Inactive status, Inactive to Active status, a change to the list of organs for which the patient is waiting, changes in patient characteristics, death, or removal.



Outcome: Relist — 14 Days

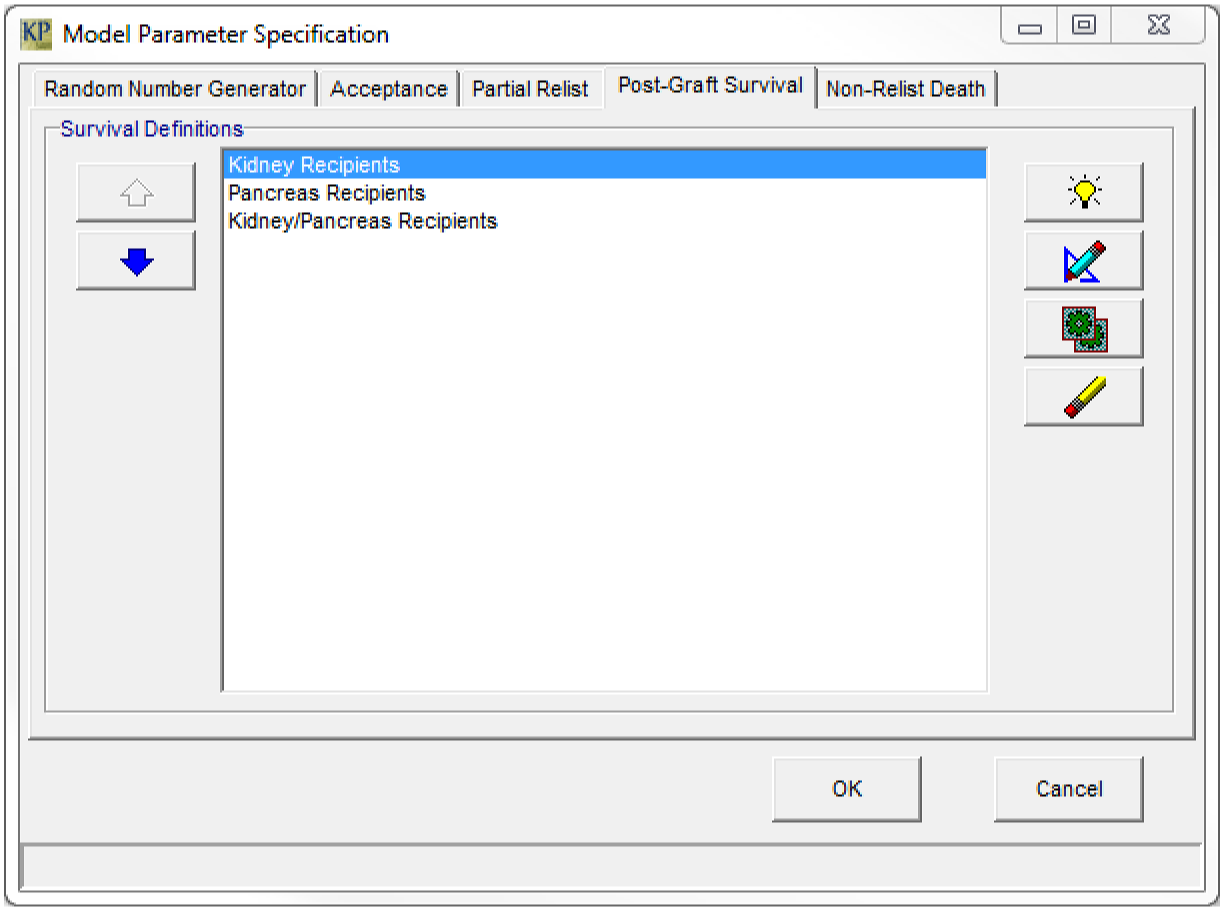
Status Updates

Days Aft	Need Kid	Need Pan	Kidney St	Pancreas	canhx_al	canhx_al	canhx_cu	canhx_cu	canhx_sr	canhx_s
0.0	T	F	1	0	.	.	.	.	.	.

Select the **OK or Cancel** button twice to return to the Model Parameter Specification panel, then choose the **Post-Graft Survival** tab to continue.

Model Parameter Specification: Post-Graft Survival

Post-graft survival time and intermediate patient status changes are determined by model input. The mechanism for specifying these post-graft survival parameters is described in detail below.



Kidney Recipients
Pancreas Recipients
Kidney/Pancreas Recipients

Highlight one of the survival definitions and select the edit button.

The Calculation Name and Patient Group tabs on the next screen are similar to the Rule Name and Rule Group tabs in the Allocation Rule Specification. Refer to the Allocation Rule Specification, page 12, for more information on those two tabs.

Select the **Covariates** tab to continue the post-graft survival definitions.

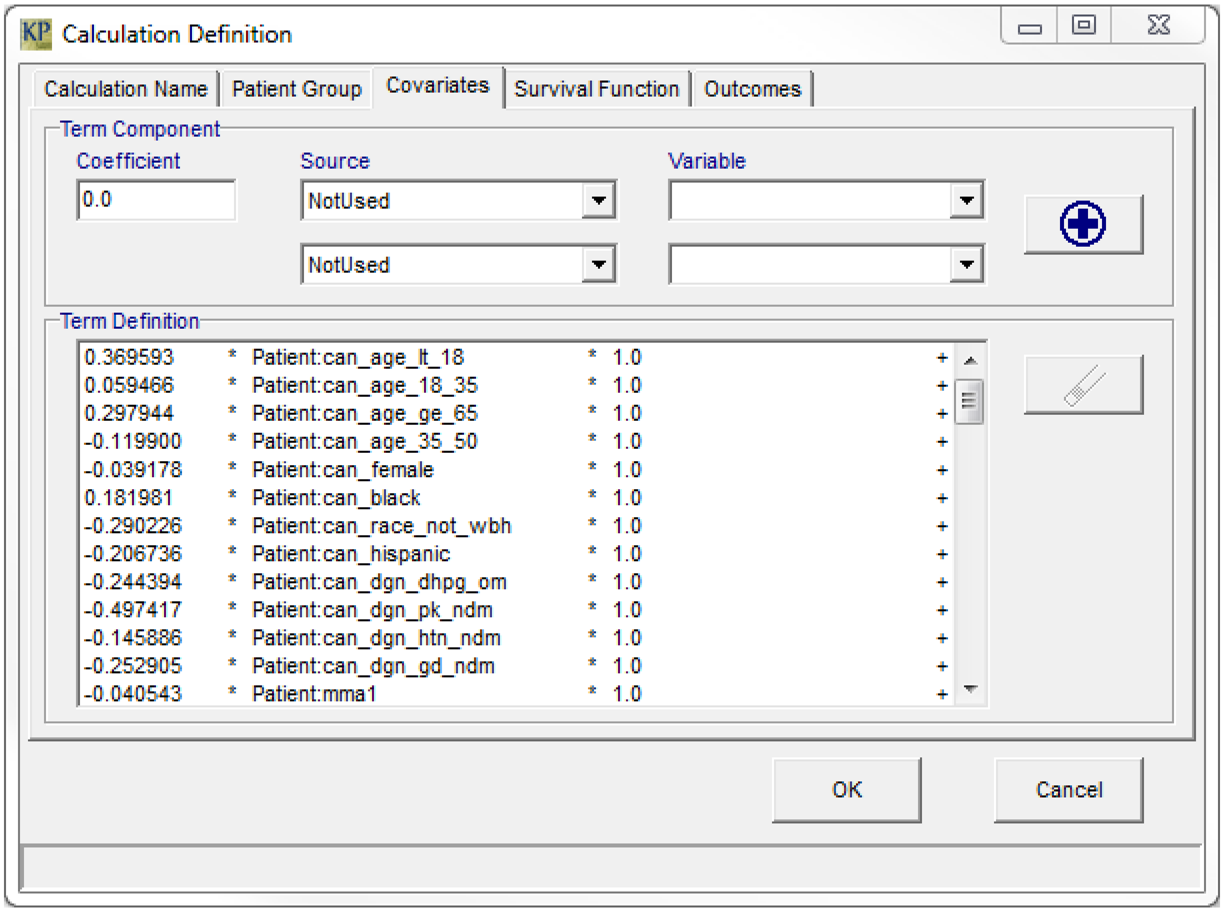
Calculation Definition (Post-Graft Survival)

Covariates

The post-graft survival equations are specified by the user. The user defines an equation for each patient group defined. This equation represents either the placement into a step table from a Cox model, or the calculation of a Weibull function.

The equation, similar to the acceptance equations, may be made up of scalar variables, organ characteristics, patient characteristics, and/or calculations that depend on information from both the organ and the patient. Each time a transplant is performed in the allocation run, this equation is solved, and the resulting value is combined with a random number; this result is then used to determine the organ failure date.

The information from the covariates screen is used to determine the relative expected survival time between patients. The actual estimated survival time is calculated based on the survival model. The survival model is made up of the covariates and a survival function. A survival function is an overall curve, describing the basic pattern of survival for patients with regard to the proportion of patients alive at a certain time. This curve is modified for each patient through the individual covariate pattern; patients with covariates indicating poor health will tend to die sooner than patients with covariates indicating good health.



Coefficient: 0.0
Source: NotUsed
Source: NotUsed

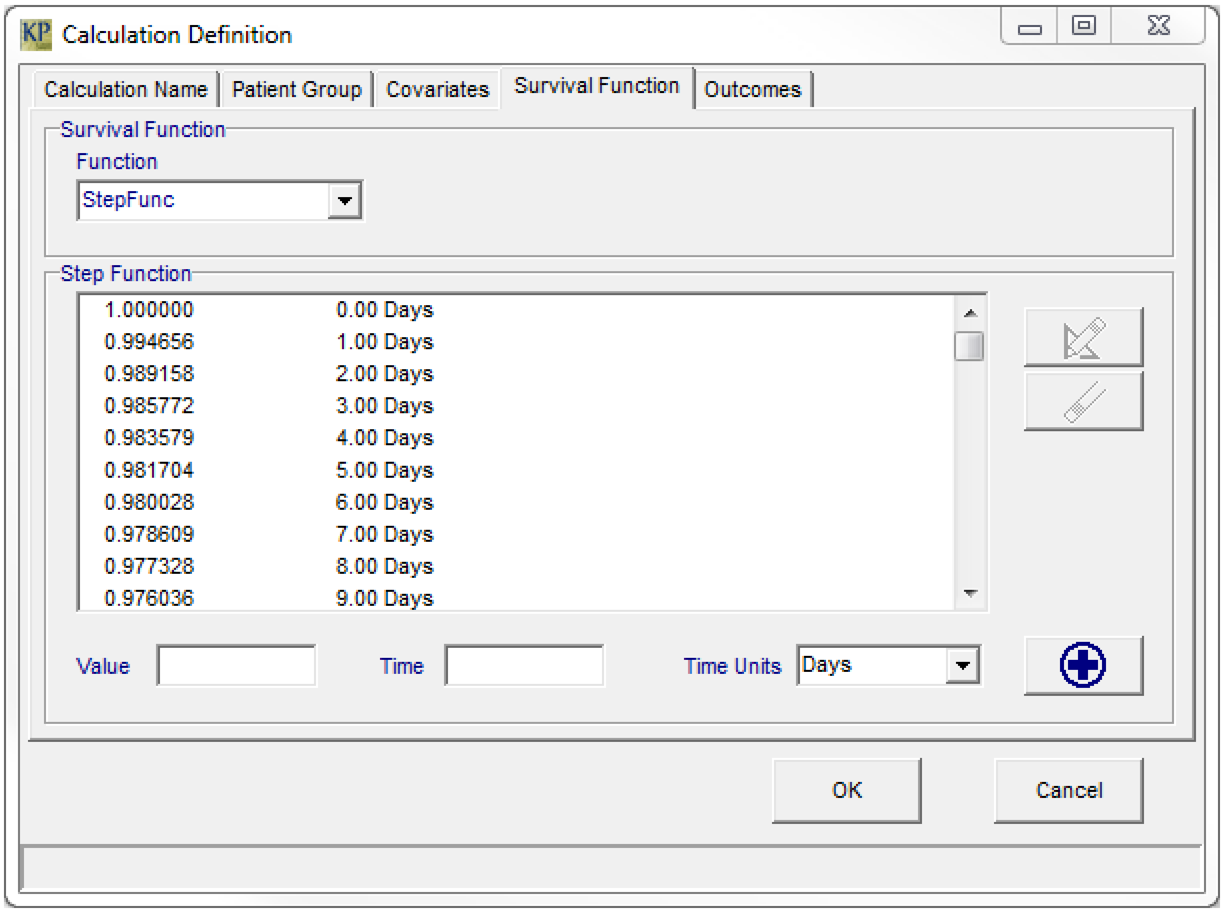
```
0.369593   * Patient:can_age_lt_18      * 1.0  +
0.059466   * Patient:can_age_18_35      * 1.0  +
0.297944   * Patient:can_age_ge_65      * 1.0  +
-0.119900  * Patient:can_age_35_50      * 1.0  +
-0.039178  * Patient:can_female         * 1.0  +
0.181981   * Patient:can_black          * 1.0  +
-0.290226  * Patient:can_race_not_wbh   * 1.0  +
-0.206736  * Patient:can_hispanic       * 1.0  +
-0.244394  * Patient:can_dgn_dhpg_om    * 1.0  +
-0.497417  * Patient:can_dgn_pk_ndm     * 1.0  +
-0.145886  * Patient:can_dgn_htn_ndm    * 1.0  +
-0.252905  * Patient:can_dgn_gd_ndm     * 1.0  +
-0.040543  * Patient:mma1               * 1.0  +
```

Select the **Survival Function** tab to continue.

<u>Calculation Definition (Post-Graft Survival)</u>

Survival Function – Cox Model Step Function

On this tab, you may choose between a Cox model step function or a Weibull distribution to determine the graft failure date. If the Cox model is chosen, then steps must be entered indicating the post-graft survival steps and associated failure times. Refer to Chapter 4, Modeling Post-Graft Survival, Graft Failures, and Relistings for more information about post-graft survival input.

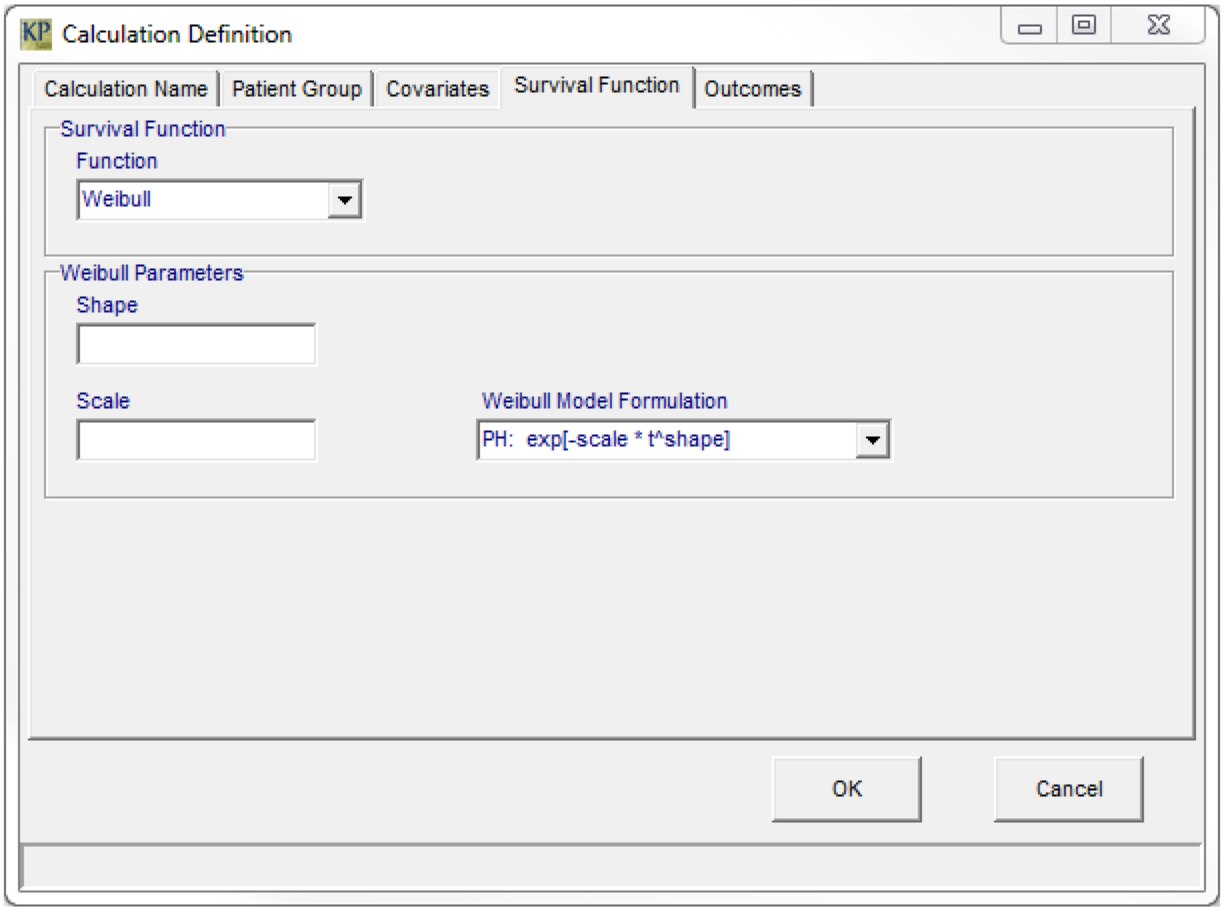


1.000000	0.00 Days
0.994656	1.00 Days
0.989158	2.00 Days
0.985772	3.00 Days
0.983579	4.00 Days
0.981704	5.00 Days
0.980028	6.00 Days
0.978609	7.00 Days
0.977328	8.00 Days
0.976036	9.00 Days

Calculation Definition (Post-Graft Survival)

Survival Function – Weibull Function

If a Weibull function is chosen, you must enter the type of formulation you would like to use for the Weibull equation. Each type of formulation requires that you enter two parameters – either shape and intercept, or shape and scale. Please refer to Chapter 4, Modeling Post-graft Survival, Graft Failures and Relistings for more information about post-graft survival input.


Calculation Definition

Calculation Name | Patient Group | Covariates | Survival Function | Outcomes

Survival Function

Function: Weibull

Weibull Parameters

Shape:

Scale:

Weibull Model Formulation: PH: exp[-scale * t^shape]

OK | Cancel

Choose the **Outcomes** tab to continue.

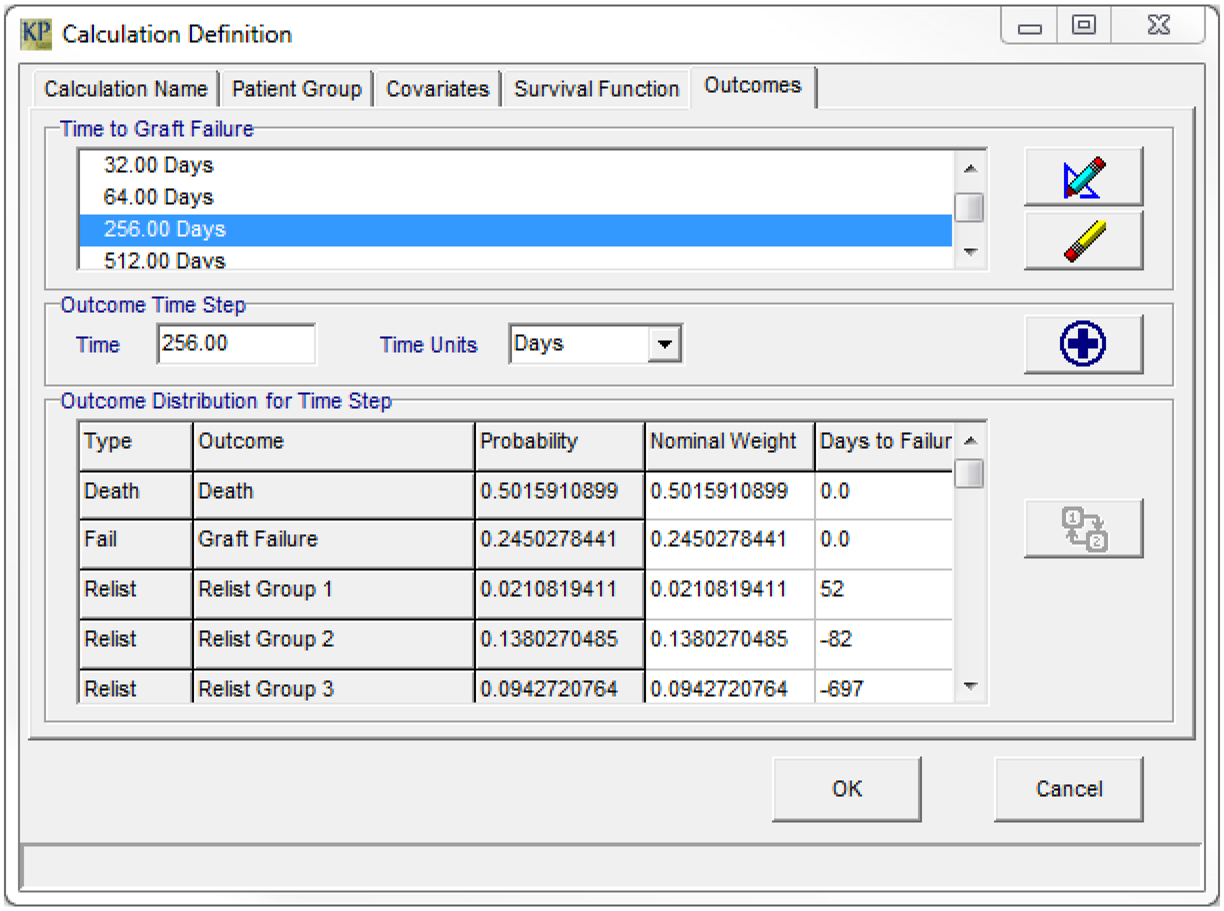
<u>Calculation Definition (Post-Graft Survival)</u>

Outcomes

Each row in the Time to Graft Failure section represents one step in the step table, and an associated organ failure date (which is equivalent to death date). Here, we will associate a set of outcomes with each step. Each outcome has a probability, also defined by the user. To see the Outcome Distribution for a particular step, highlight the step, then select the edit button to the right of the Time to Graft Failure section.

Each outcome has a probability, and a number of days before failure when this outcome will occur. In addition, if the type of outcome is relist, one or more status updates may occur between relisting and organ failure. These are also defined by the user.

For example, suppose a patient undergoes transplant and arrives at this outcome step table with a failure time of 256 days after transplant. Another random number is drawn, and the patient is assigned an outcome from the list at the bottom of the screen. If this patient ended up in Relist Group 1, then at 52 days before organ failure, this patient would relist. The patient may also have status updates, described below.


Calculation Definition

Calculation Name | Patient Group | Covariates | Survival Function | Outcomes

Time to Graft Failure
- 32.00 Days
- 64.00 Days
- 256.00 Days
- 512.00 Days

Outcome Time Step
Time: 256.00 Time Units: Days

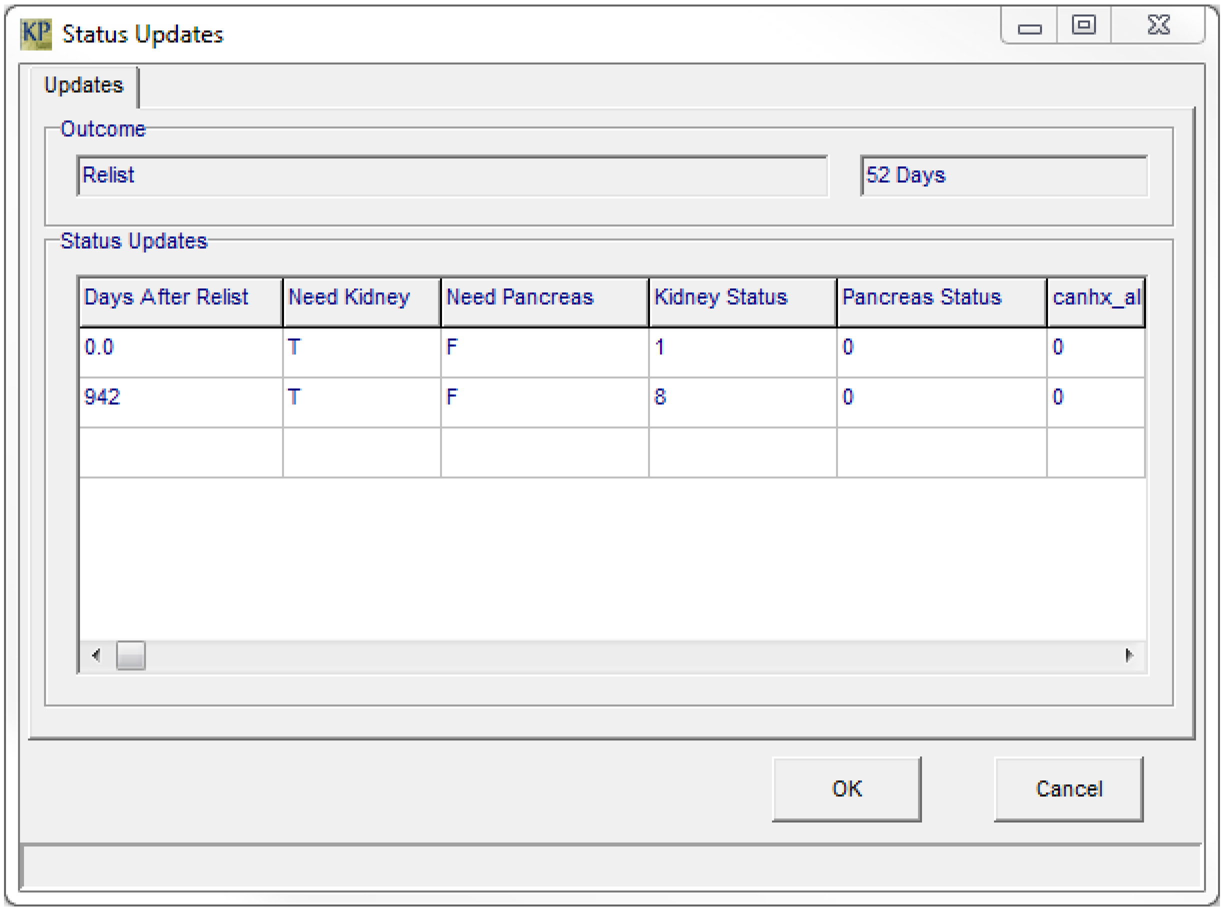
Outcome Distribution for Time Step

Type	Outcome	Probability	Nominal Weight	Days to Failur
Death	Death	0.5015910899	0.5015910899	0.0
Fail	Graft Failure	0.2450278441	0.2450278441	0.0
Relist	Relist Group 1	0.0210819411	0.0210819411	52
Relist	Relist Group 2	0.1380270485	0.1380270485	-82
Relist	Relist Group 3	0.0942720764	0.0942720764	-697

Highlight Nominal Weight for a Relist outcome, and select the update button (squares and arrows) on the right to continue.

1.6.5 Status Updates

All status updates that you would like for this patient to receive subsequent to transplant should be defined on this screen. Each update is relative to the date on which the patient is scheduled to relist, which is relative to the organ failure date. The status updates definitions are similar to those entered in Partial Relist, above.



Outcome

Relist | 52 Days

Status Updates

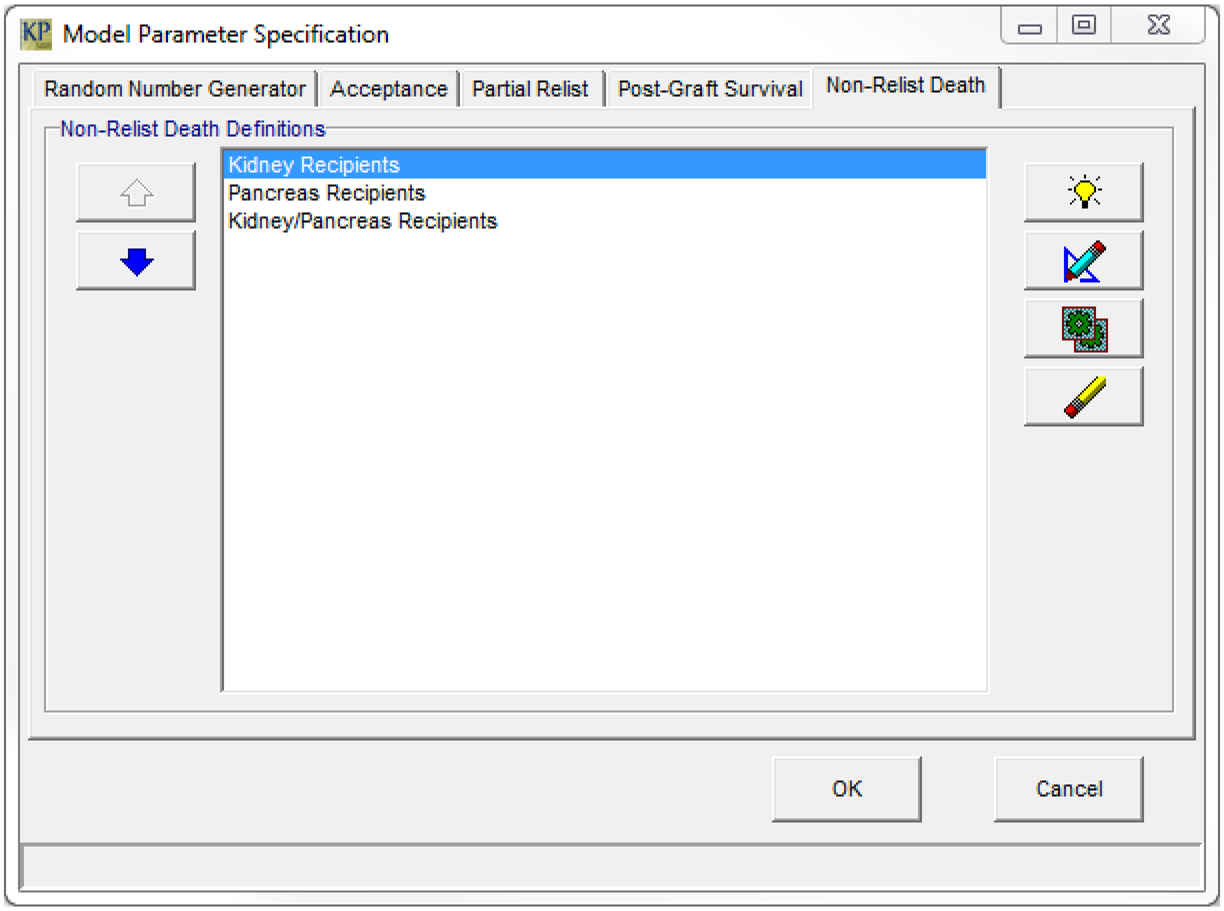
Days After Relist	Need Kidney	Need Pancreas	Kidney Status	Pancreas Status	canhx_al
0.0	T	F	1	0	0
942	T	F	8	0	0

Please select the OK or Cancel button twice to return to the **Model Parameter Specification panel**, and then choose the **Non-Relist Death** tab to continue.

Model Parameter Specification: Non-Relist Death

The non-relist death panel contains the definition of calculation to perform to determine how long after graft failure a graft recipient who does not relist will live.

The panels within Non-Relist Death are identical to those within Post-Graft Survival. Please refer to the Post-Graft Survival section of this chapter for more information.


Model Parameter Specification

Random Number Generator | Acceptance | Partial Relist | Post-Graft Survival | Non-Relist Death

Non-Relist Death Definitions

Kidney Recipients
Pancreas Recipients
Kidney/Pancreas Recipients

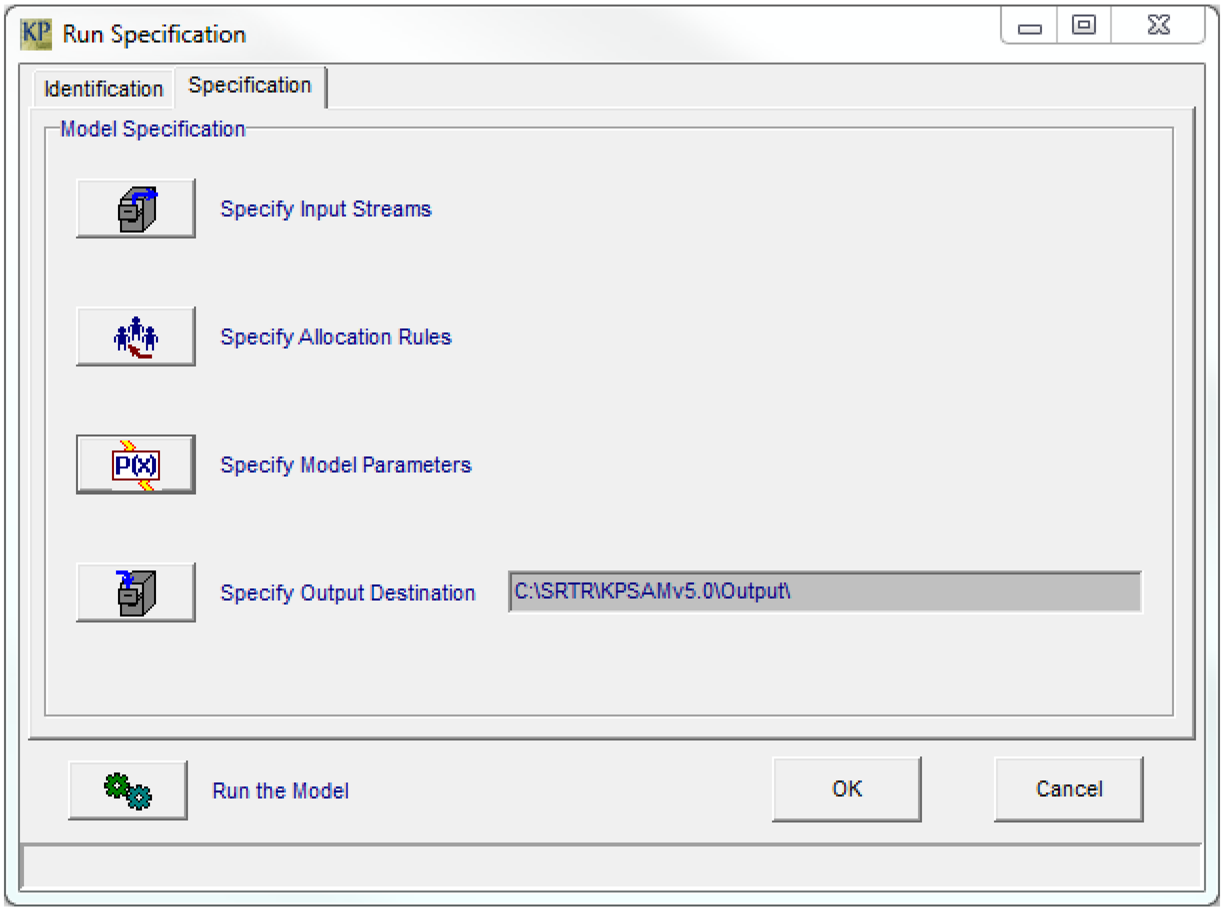
Select the **OK or Cancel** button to return to the **Run Specification** panel to continue.

1.6.6 Output Destination

Run Specification

Specification

The final step is to specify the destination of outputs. The current output destination appears next to the Specify Output Destination button of the Specification tab.



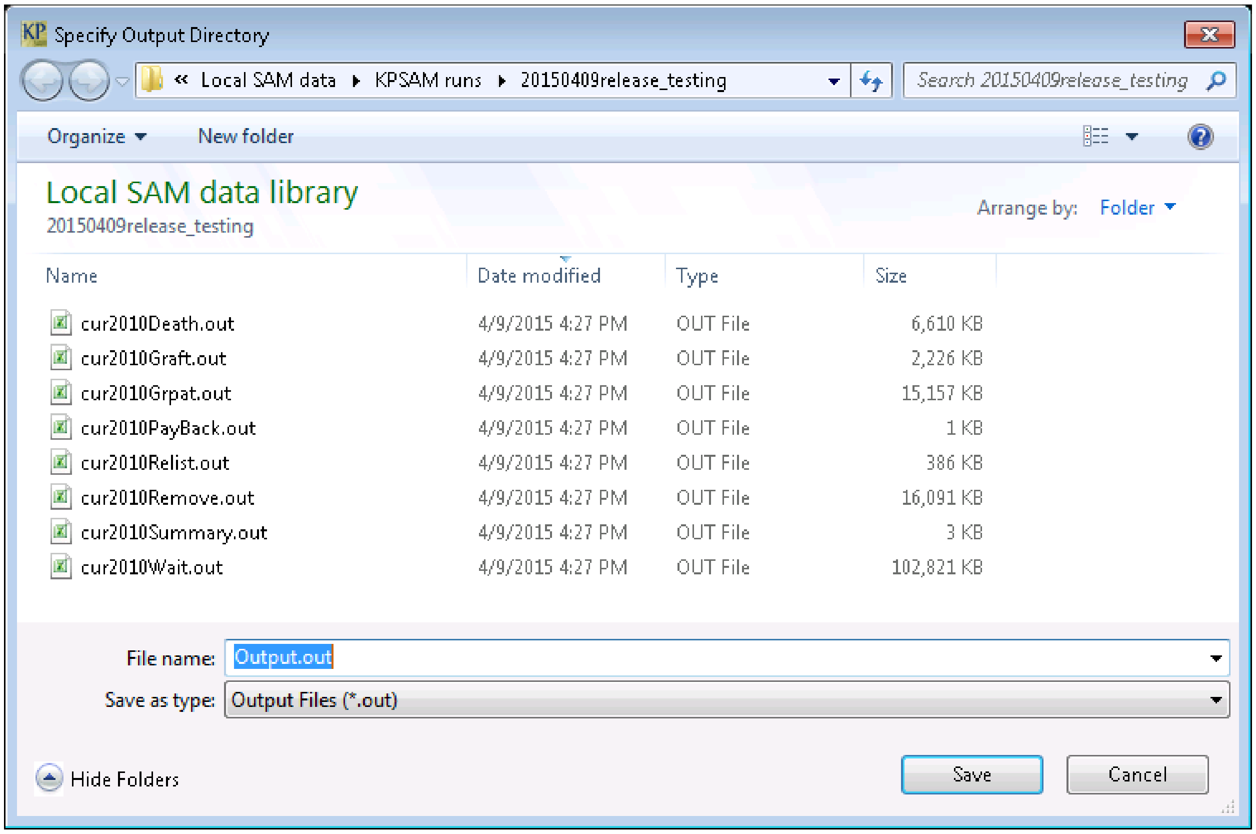
C:\SRTR\KPSAMv5.0\Output\

Select the Specify Output Destination button, and the Specify Output Directory dialog appears.

Specify Output Directory

The simulation creates output files for each Allocation Run in the selected directory. The file names are those used in the source code. The model assigns each output file a different name by prefixing the model short name of the Allocation Run. See Chapter 8, Frequently Asked Questions, for more details.

This screen operates with the usual conventions of a Windows save dialog. The process for selection or creation of an output folder should be familiar to Windows users.


Specify Output Directory

« Local SAM data ▸ KPSAM runs ▸ 20150409release_testing

Local SAM data library
20150409release_testing

Name	Date modified	Type	Size
cur2010Death.out	4/9/2015 4:27 PM	OUT File	6,610 KB
cur2010Graft.out	4/9/2015 4:27 PM	OUT File	2,226 KB
cur2010Grpat.out	4/9/2015 4:27 PM	OUT File	15,157 KB
cur2010PayBack.out	4/9/2015 4:27 PM	OUT File	1 KB
cur2010Relist.out	4/9/2015 4:27 PM	OUT File	386 KB
cur2010Remove.out	4/9/2015 4:27 PM	OUT File	16,091 KB
cur2010Summary.out	4/9/2015 4:27 PM	OUT File	3 KB
cur2010Wait.out	4/9/2015 4:27 PM	OUT File	102,821 KB

File name: Output.out
Save as type: Output Files (*.out)

The example lists the output files from an Allocation Run named cur2010. You also may find it useful to create new folders to hold the inputs and outputs of different sets of Allocation Runs.

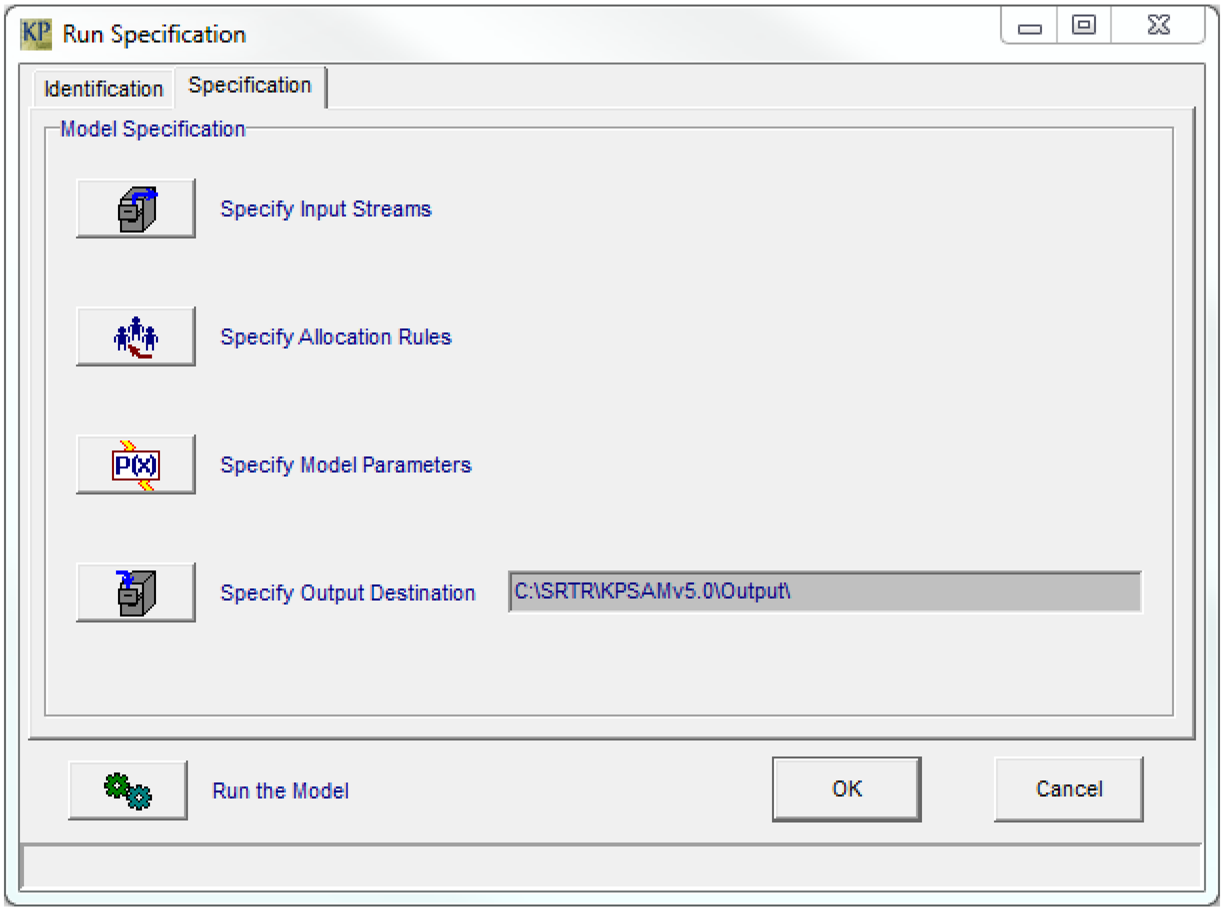
Select **Save** to return to the **Allocation Run Specification** screen.

1.6.7 Running the Model

We have visited the screens that specify the input data for the model and have returned to the Run Specification screen.

Run Specification

Now all the Allocation Run parameters are set, and we can run the model.



C:\SRTR\KPSAMv5.0\Output\

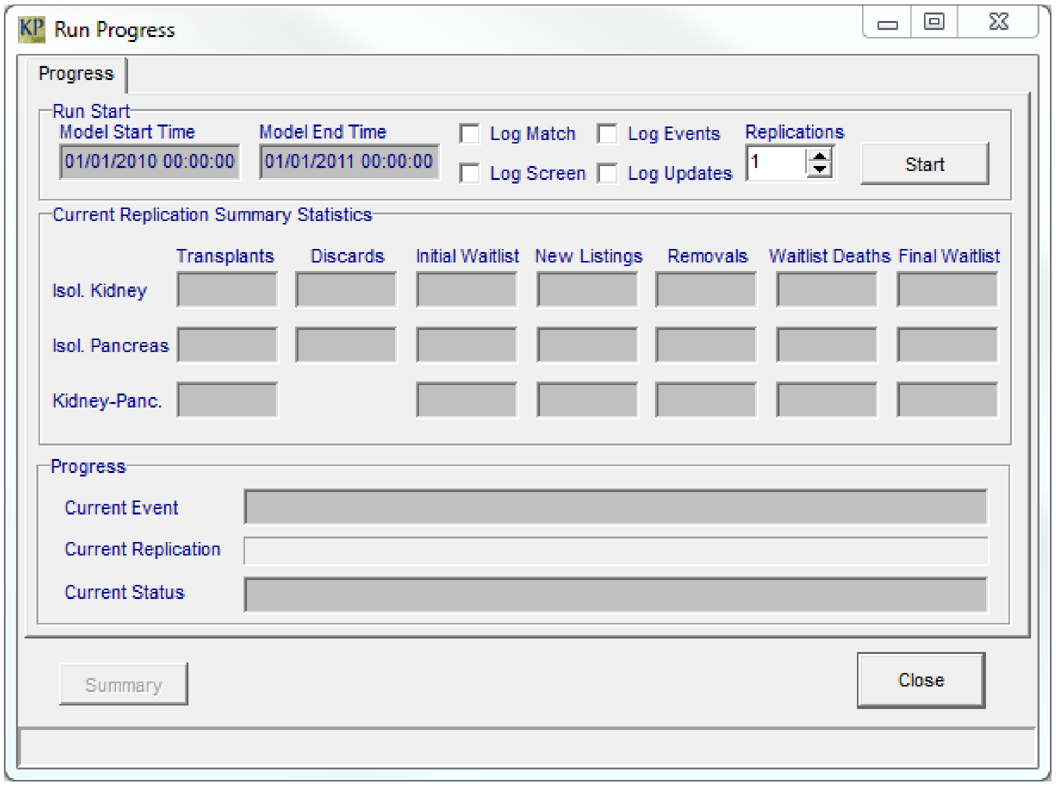
Select the **Run the Model** button. The Run Progress screen appears.

Run Progress

On this screen, you can select the number of replications to run and, optionally, you can choose to Log Match, Log Events, and/or Log Updates. You can also choose to log the organ offers that are made by marking the box next to Log Match. Please note that the match offers and patients screened log files can get quite large, so if you are running multiple replications, you may want to avoid logging the match offers and screened patients. See Chapter 7, Output File Specifications, for more details about these options.

Select **Start**. You will see the summary statistics and progress bars updated periodically as the Allocation Run progresses. While the model is processing, the Start button changes to a Stop button. This permits you to interrupt the Allocation Run process if necessary. The Start/Stop button changes to a faded Done message and the total running time is displayed in the Current Event box when the Model Allocation Run is complete.

The random number seed stream is initialized before the first replication. Subsequent iterations continue drawing from that stream where the previous iteration left off.

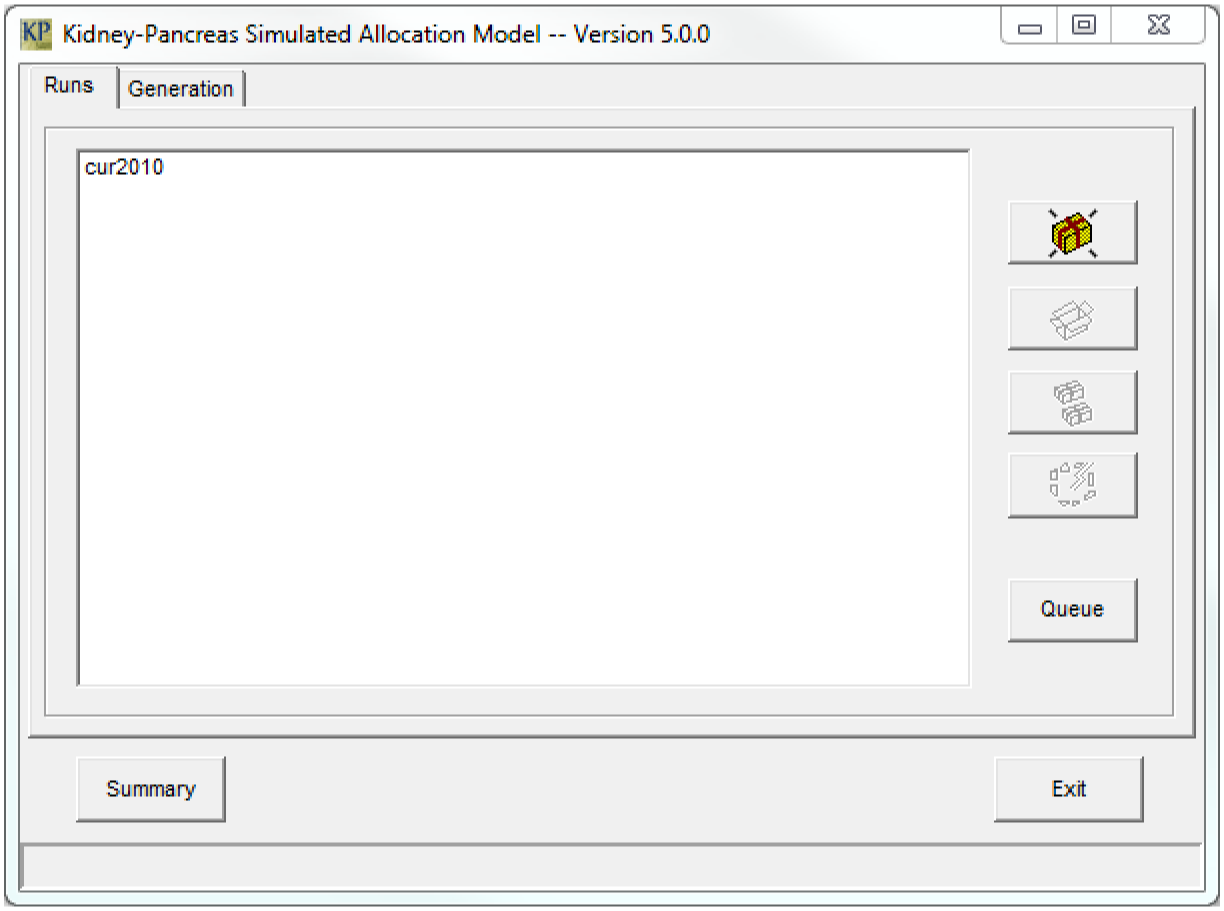


At the completion of the Allocation Run, you may view the summary statistics by selecting the Summary button. (Summary statistics also are recorded in an output file.) From the Summary screen, the Close button returns you to the Run Progress screen.

When the Allocation Run is complete, select the Close button. You will be taken to the Allocation Run Specification screen. Closing out of that screen takes you back to the KPSAM main screen.

1.6.8 Main Screen

Upon returning to the main screen, you have several options. If you have no more Allocation Runs to perform, select the Exit button, and the program will terminate.



If you have just defined a new Allocation Run package, its name will be listed on the main screen, along with the names of any previously defined packages. Start another Allocation Run by selecting one of the listed Allocation Run packages and then selecting the open box button. This will take you to the Run Specification screen, where you may run that package or edit its definition. Other button options allow you to create another new Allocation Run package, duplicate an Allocation Run package, or delete an Allocation Run package from the list.

You may also run the Queue (section 1.7), which lets you perform multiple runs without user input, and create a Summary file (section 1.8) from the available output.

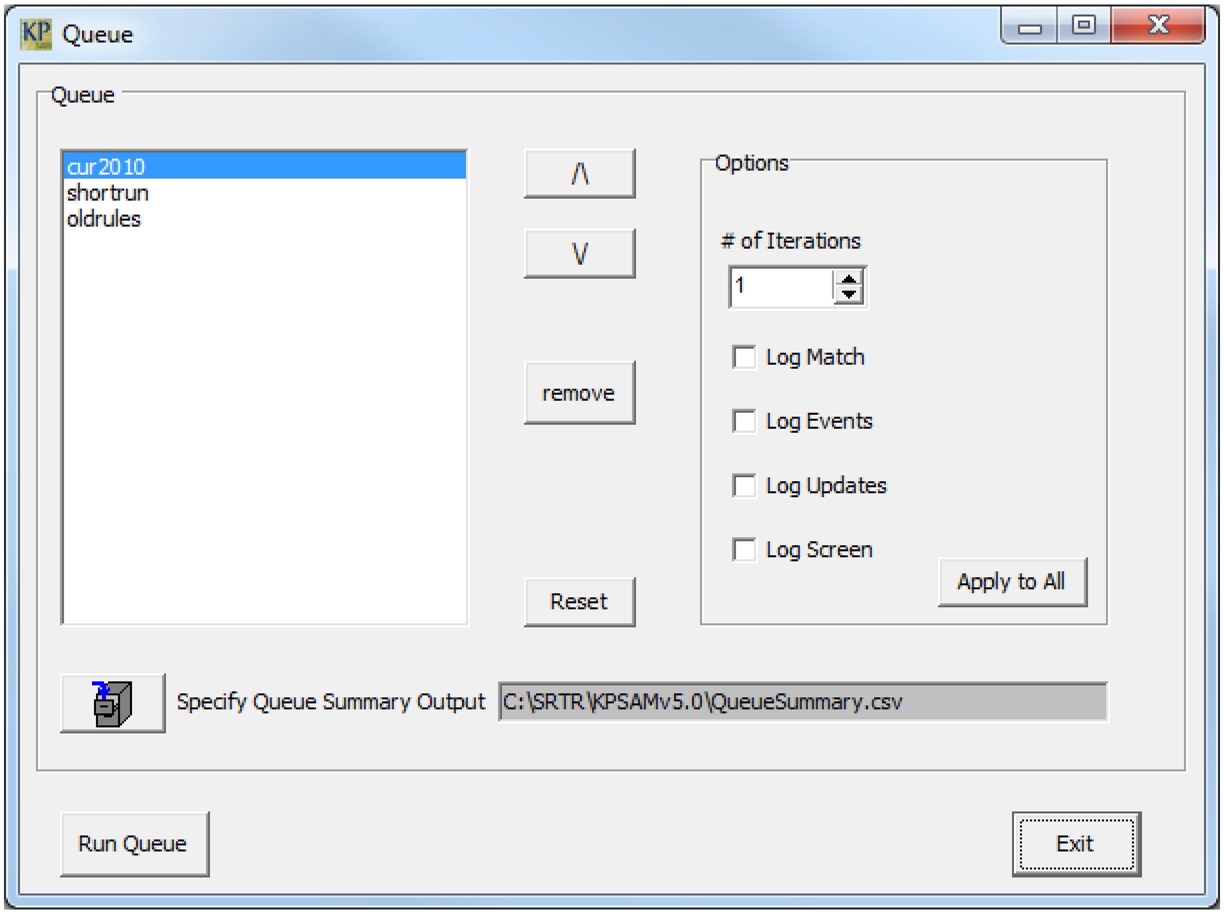
Four operations are allowed on run packages:

- Create a new Allocation Run package (wrapped package).
- Open the selected Allocation Run package (open box).
- Copy the selected Allocation Run package (two wrapped packages).
- Delete the selected Allocation Run package (lightning bolt destroying package).

1.7 Queue

The Queue feature allows users to select multiple models to run one after another without further user input. The user specifies the models, their order, and their settings, and then the program performs all the runs and returns to the Queue screen.

To open it, select the **Queue** button at the KPSAM main screen.



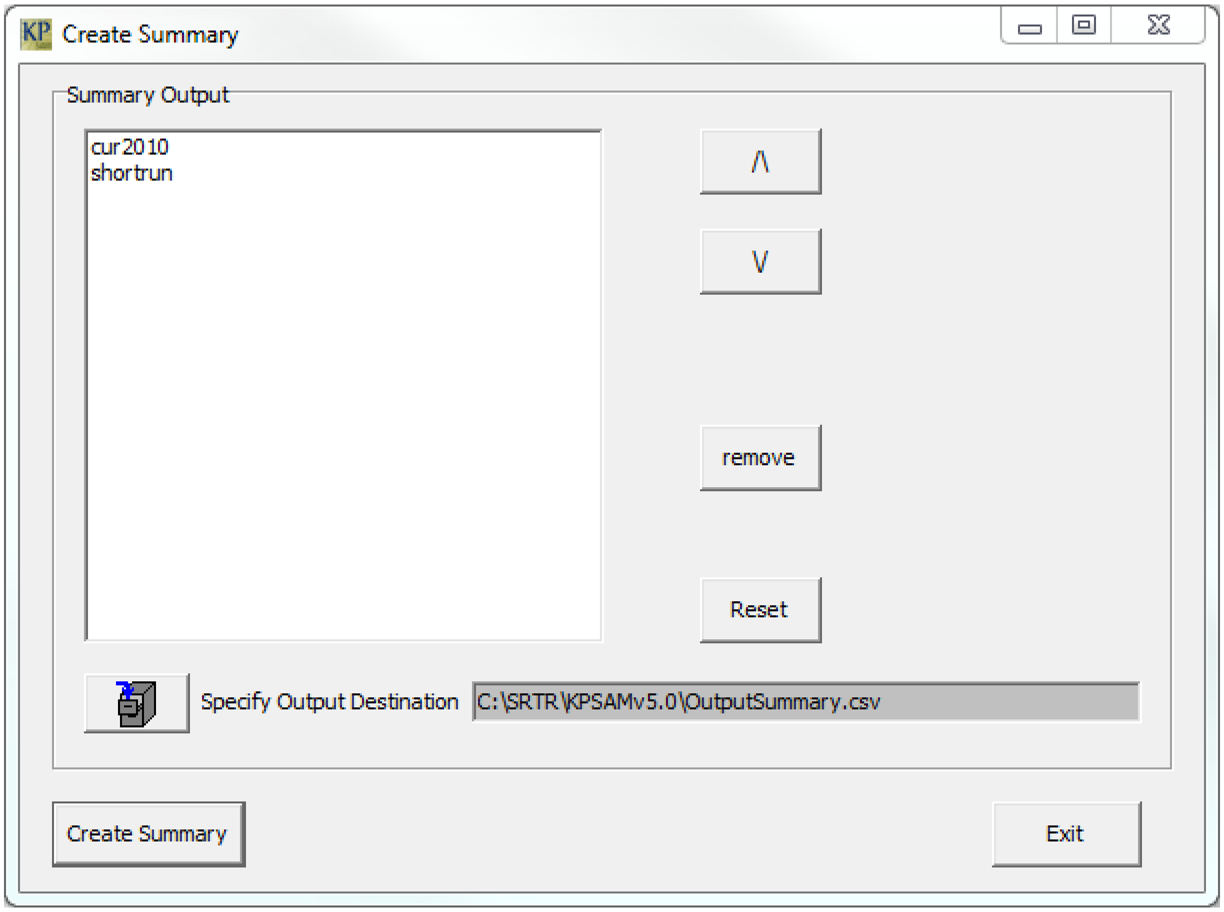
All the models are listed when the Queue is opened. Users can select the buttons in the middle to change the model order (runs go from top to bottom), remove runs, or reset the queue. Users can select options for each model on the right side, and apply the settings to all models in the queue.

After the run is completed, the Queue creates a QueueSummary.csv file, which includes all the model summary files in an Excel-friendly format.

To run the queue, select the Run Queue button on the bottom left. After the queue is completed, the program will return to the Queue screen, informing the user of the run time and if any errors occurred.

1.8 Summary

The Summary feature allows users to create an output file from any available model summaries. To open it, select the Summary button on the bottom left of the main screen of the KPAM.



The program scans the models and checks if an output summary exists in the model's specified location (for example, if you create a model and specify to place output in My Documents\KPSAM Output, the program will look for the output summary in that folder). The user will see the models for which the output summary exists.

The user can select the order, remove models or reset the list. They can also select the OutputSummary destination.

Select the Create Summary button on the bottom left to create the summary file. The file is in .csv format.

1.9 Viewing Output Files

The simulation creates output files for each Allocation Run, and it assigns file names by appending the short title of the Allocation Run to a specific identifier. For example, the file xxxGraft.out contains the list of transplants (grafts) that were performed for an Allocation Run titled "xxx."

To view the output files, you can use Excel or another spreadsheet application and open the file with the vertical bar "|" as the **delimiter**.

2 Overview

This chapter provides an overview of the modeling methods and of the way computation is organized in the organ allocation model. The discussion covers the basic modeling approach (including which processes are random in the model), the top-level organization of the model, the main data structures referenced by the model, and the event handler routines.

2.1 Basic Approach and Random Processes

The model simulates the organ allocation system with an event-sequenced Monte Carlo technique. That is, some of the modeled processes are random in nature, and the model samples pseudo-random numbers to simulate a realization of processes over the specified time period. Each such realization of the organ allocation system constitutes a single replication. The model generates a user-specified number of replications, saves the detailed histories of these replications in user-specified files, and describes the set of replications with respect to the means, medians, and standard deviations of selected variables.

Some important assumptions:

1. Arrivals of candidates are input to the model with a data file.
2. The initial wait list is input to the model with a data file.
3. An entire history of wait-list status changes (to the end of the Allocation Run, death, or removal from the wait-list) must be input to the model for each patient (active / inactive status changes, time of removal, and time of death). This is the history that will be used for the patient up until the time (if any) that the patient is allocated a transplant during the simulation. Note that this history does NOT specify the time of a transplant. This history can be based on actual experience, although a hypothetical history must be prepared for transplanted patients to tell what would have happened to them had they not received a transplant. Alternatively, the histories can be based on data generated from hypothesized models.
4. Once a candidate receives an organ, that patient's input stream of status changes no longer applies. If the patient relists, the model assigns a status change history to the patient by randomly selecting a set of status changes from a pool of user-defined histories specifically provided for this purpose.
5. The values of several other parameters are specified in the program or tables and remain constant during a run. These include the parameters of the graft failure time distribution and the geographic membership relationships among institutions, local units (OPOs), and regions. For example, it is assumed that patients do not move among institutions, and institutions do not change affiliation with OPOs. Because

these parameters and relationships are controlled by input data, they can vary from case to case.

As the list shows, the model represents several important processes – candidate and organ arrivals and changes of status – through user input. Sample histories of these processes must be created outside the model (e.g., using historical data) and formatted into time-ordered streams of records for input to the model.

Other processes are represented internally to the model. The following processes are simulated within the model using Monte Carlo techniques, i.e., a pseudo-random number generator:

1. **Organ acceptance:** The user defines a calculation used to compute the organ acceptance probability. Values that may be used in this calculation are scalar variables, characteristics of the organ or donor, characteristics of the potential organ recipient, and/or values calculated from characteristics of the specific organ and patient combination under consideration. This calculation is performed for each patient to whom the organ is offered, and the resulting value, β, is transformed using an inverse logit transformation (exp(β) / 1+exp(β)). That value is then compared to a random number between 0 and 1. If the random number is less than this value, then the organ is accepted; otherwise, it is refused. An organ is considered discarded after it has been offered to all potential recipients and each offer has been refused, or after it reaches the maximum organ offer count specified on the Acceptance Definitions screen.

2. **Post-graft survival:** Post-graft survival is also specified in the model by the user. The user defines a calculation that is then used to determine the patient's death date after transplant. The calculation may be made up of scalar variables, organ characteristics, candidate characteristics, and/or calculations that depend on information from both the organ and the candidate. Each time a transplant is performed in the model, this calculation is performed, and the resulting value is combined with a random number; the result of this is used to determine the death date, using the method chosen by the user, either a Cox proportional hazard model or a Weibull distribution. A set of possible outcomes and their relative probabilities is associated with each death date. The possible outcomes are relisting at differing times before the death date and with differing medical characteristics, or not relisting.

Here, a relisted graft recipient is one who received a graft during the time span of the simulation, not necessarily one who entered the model with a prior graft. The latter set of recipients enter the model on the arrival stream or on the initial waiting list, and valid records presumably exist for them in the status change input stream (until the simulation awards new grafts to them).

The model draws all random numbers from a single random number stream.

2.2 Top-Level Overview: Event Queues

The simulation maintains time-ordered queues of several kinds of events, from which it can pick the next event in order for processing:

1. Organ arrivals (input-driven)
2. Status changes for wait list candidates who have not yet received grafts (input-driven)
3. Unacceptable antigen changes for wait list candidates who have not yet received grafts (input-driven)
4. Patient arrivals (input-driven)
5. Status changes for relisted graft recipients (sampled from a pool of possible outcomes)
6. Relisting events of recipients whose grafts fail (sampled by the model)
7. Deaths of graft recipients not on the wait list (sampled by the model)

Note that status change events include changes that indicate the patient have been removed from the waitlist or has died while on the waitlist.

2.3 Event Handlers

2.3.1 Organ Arrival Event

This event handler selects a candidate to receive the organ that has become available. It applies the allocation rules that have been defined in the model. It performs the match run by reordering the wait list according to the rules and offering the organ to candidates in order. It simulates the organ acceptance process by sampling from a uniformly distributed random variable and comparing this to the probability of acceptance, which is calculated using acceptance inputs to the model. If no candidate accepts the organ, the organ is discarded. If the organ is accepted, the event handler removes the recipient from the wait list. Whatever the outcome, the event handler writes a record with the results of the match run.

The event handler places the recipient on a list of graft recipients and schedules a graft failure event for the recipient. It uses the model (as described in Chapter 4) to determine a possible outcome prior to graft failure and the time that will elapse until this outcome. It then schedules the outcome events, which could include a relisting and possibly some post-graft status change events. Please note that these status change events are defined differently in the model from those that take place prior to transplant. The latter are described next.

2.3.2 Status Change Event

The status change file contains records that describe the medical status history of every wait list candidate from the time of the candidate's arrival to the model (either the initial wait list snapshot or arrival to the wait list) until the candidate's death. This history is valid until such time as the candidate may receive a graft. If a transplant recipient relists, the recipient's status history is provided by a different source. Whichever source of status changes applies, the model invokes the status change event handler when a status change event occurs.

If the candidate's status has changed from active to inactive or vice versa, the model updates variables that keep track of the time that the candidate has been in the previous status. The event handler updates the candidate's status to the new value.

If the new status is 9 (removal), the event handler removes that candidate from the wait list and places the candidate on a list of removed patients. If the new status is 8 (death), the event handler removes the candidate from the wait list and writes an outcome record for the patient. If the patient dies after removal from the wait list, the model removes the patient from the list of living removed patients.

2.3.3 Candidate Unacceptable Antigens Change Event

The candidate unacceptable antigens file contains the list of donor antigens, by locus, that are not acceptable to this candidate. This list can change over time, so each of these entries is time-stamped. Whenever a change to this list occurs for a candidate, the unacceptable antigens update event handler is invoked. All updates with the same audit identifier constitute a set. The old set of antigens for that candidate is discarded and the new set is applied.

2.3.4 Candidate Arrival Event

This event occurs when a patient joins the wait list. When this happens, the event handler places the patient on the wait list and initializes all descriptions of the candidate, e.g., demographic descriptors, previous transplantation status, and institution where listed. The specifications of the candidate arrival record appear in Chapter 6 of this guide.

2.3.5 Post-Transplantation Events

When a patient receives a graft (i.e., at the time of an organ arrival event), the simulation samples the future time of graft failure and determines whether the patient will relist or not. The simulation accordingly schedules the patient’s death and, potentially, the relist event. The event handler for post-transplantation events processes these events.

In the case of a death event, the event handler removes the patient from the list of living graft recipients and writes an outcome record. In the case of a relisting event, the simulation removes the patient from the list of non-wait-listed graft recipients and adds the patient to the wait list. A status change history was already selected for the patient at the time of the organ arrival event, and the event handler initializes the patient’s status, and the organ(s) for which they are listing to the initial set of values provided in this history.

2.4 Limitations

As discussed at the beginning of this chapter, KPSAM relies on the inputs selected at runtime to simulate organ allocation. Any bias or omissions in the input data may affect simulation results. In addition to this general caveat, several specific limitations are worth noting when using KPSAM and interpreting its results.

2.4.1 Reliance on Historical Data

KPSAM uses statistical models to simulate multiple parts of the organ allocation process. These models were trained on historical data from OPTN and SRTR transplant archives; thus, they may not respond to changes in allocation policy in the same way the real system would. This is particularly true of the organ acceptance model, which simulates the decision-making behavior of patients and clinicians based on historical acceptance data. KPSAM also does not identify trends over time in the training data, so gradual changes in the characteristics of donors and candidates will not be reflected in KPSAM results.

2.4.2 Standardized Behavior

The OPTN allocation guidelines are extensive, but they do not eliminate all ambiguity. In practice, there is some amount of variation in policy and behavior between different OPOs and transplant centers. However, KPSAM assumes that all centers and OPOs implement allocation policies in the same way and exhibit the same organ acceptance behavior. KPSAM also does not model any directed or expedited allocation of donated organs.

2.4.3 Status Matching and Subgroups

Patients who underwent transplant in real life have no status updates in their patient records past the transplant date, so status updates from other patients were appended to fill out their KPSAM patient histories in the SRTR-provided status input file. These patients are matched on expected mortality but not on specific diagnosis, so KPSAM's ability to predict outcomes in particular diagnosis subgroups may be limited.

2.4.4 Listings versus Patients

KPSAM models listings rather than patients. A patient will appear in the KPSAM input files If a patient is wait listed at multiple centers, this patient will appear several times in the KPSAM input files.

2.4.5 Discards

Organs are discarded after a fixed number of declined offers, regardless of organ and donor characteristics.

3 Kidney and Pancreas Allocation Rules

The model is sufficiently flexible to allow the user to specify a wide range of allocation policies through the input files that are provided specifically for that purpose. The model is also distributed with two sets of input files that implement (1) the current Kidney Allocation System, as of December 2014, and (2) the Kidney Allocation System in effect prior to December 2014. The most current rules, described by OPTN policies 8 and 11, are available at http://optn.transplant.hrsa.gov/governance/policies/; historical overviews and summary flowcharts can be downloaded at http://srtr.org/allocationcharts/Default.aspx. Details for the implementation of some specific policy elements in KPSAM are described below.

3.1 Kidney Point System

Patients are awarded points based on waiting time, number of HLA mismatches, cPRA, and child and adolescent patients receiving offers from donors with KDPI <0.35. All points are awarded via the KPSAM score boosting mechanism. The number of points awarded for mismatches are 2 and 1, respectively, for 0- and 1- DR mismatches. Please refer to the score boost definitions in the KPSAM Input Files Guide for more information.

3.2 Screening Criteria

The rules screen out candidates from consideration for any one of the following reasons:

1. **ABO:** donor and candidate types incompatible.
2. **Age:** donor age not in candidate's acceptable age range
3. **Weight:** donor weight not in candidate's acceptable weight range
4. **HCV status:** donor is HCV positive, but candidate has indicated they won't accept an HCV positive donor
5. **Candidate status:** candidate's status is temporarily inactive
6. **Mismatch unacceptable:** candidate's HLA mismatch with this donor is higher than the candidate's specified maximum for one or more combinations of HLA.
7. **Unacceptable antigens:** donor has one or more antigens listed as unacceptable for this candidate.

Status

The following table summarizes the medical statuses for all patients. Each patient has a status field that relates to their kidney waitlist status, and one that relates to their pancreas waitlist status. For the model, medical status incorporates both the active/inactive status, removal and waitlist death codes from the SRTR data into one field.

Status	Definition
0	Patient is not waiting for this organ.
1	Actively waiting for transplant.

7	Patient is temporarily inactive. Waiting time will not be accrued by patients registered as inactive.
8	Patient has died while waiting for a transplant.
9	Patient has been removed from the waiting list (for reason other than living donor transplant) and is no longer considered a part of the system.
15	Patient has been removed from the waiting list because they received a living donor transplant.

3.3 ABO Typing

Organs will be allocated to patients according the following blood type compatibilities:

	Candidate's Blood Type			
Donor	**O**	**A**	**B**	**AB**
O	I	C/X	C/X	C/X
A	X	I	X	C
B	X	X	I	C/X
AB	X	X	X	I

I: Treated as Identical for purposes of allocation
C: Compatible
X: Incompatible
C/X: Considered compatible for zero-ABDR mismatch patients, and incompatible for all others

4 Modeling Post-Graft Survival, Graft Failures, and Relisting

For organ allocation, post-transplant survival is an important factor to consider. This section discusses how the simulator, based on model inputs, estimates how long a patient would live should that patient be given a particular organ at a particular time. This can be used to calculate, after a series of model runs, whether a given policy would result in greater organ utility as measured by overall post-transplant survival.

4.1 Time to Graft Failure Determined with a Cox Proportional Hazards Model

This section describes the process for entering the parameters required for KPSAM to determine the time to graft failure using a Cox proportional hazards model.

When a candidate receives a graft, the model samples the time, T, remaining before death. The user specifies the survival model by specifying the model variables and the coefficients of the following survival function:

$$\text{Prob [survival time > T]} = S_j(T)^{\exp(\Sigma b_{ij} x_{ij} y_{ij})}$$

Where:

$Sj(T)$ is a step function specified by the model user,

j is an indicator for recipient group as specified by fields in the recipient and organ records. The user may select a different set of variables and coefficient values for each group, e.g., diagnosis-specific survival models,

b_{ij} is the ith coefficient within group j, i.e., $\Sigma b_{ij}x_{ij}y_{ij}$ = the sum of the product bxy over all covariates (i) within group j,

x_{ij} is either the ith covariate within group j or the first part of an interaction term within group j,

y_{ij}, if specified, is the second part of an interaction term within group j.

Covariates and Parameters

Each element x_{ij} and y_{ij} is the value of a model variable or of unity. (A model variable here means the value of a variable describing the recipient or the organ involved in the graft event.) The user selects the variables x_{ij} and y_{ij}, and provides the corresponding coefficients b_{ij}. This is done separately for each recipient group.

$x_{ij} = 1$ and $y_{ij} = 1$ implies that this is the intercept term.
x_{ij} = variable and $y_{ij} = 1$ implies that this is an ordinary term within the model.

x_{ij} = variable1 and y_{ij} = variable2 implies that this is an interaction term between
variable1 and variable2.

Survival Time

At each transplant event, the model determines the remaining survival time, T, by sampling a value u from a U(0,1) distribution and inverting the complementary cumulative probability distribution of the survival time for this recipient:

$$\text{Prob [survival time > T]} = S_j(T)^{\exp(\Sigma b_{ij} x_{ij} y_{ij})} = u$$

$$T = S_j^{-1}(\exp(\ln(u)/\exp(\Sigma b_{ij} x_{ij} y_{ij})))$$

Step Function

The model reads data for $S_j(t_{kj}) = v_{kj}$ in tabular form (separately for each group j). These are specified by the user in the Default Survival input file, described in Chapter 6, or on the Step Function screen, described in Chapter 1, section 6.4. Specifically, the table includes values of t increasing from t=0 (time of transplant) to t=maxtime, and corresponding values of v declining from v=1 (i.e., all recipients are alive at time zero) at t=0 to v=0 for the largest t (eventually all the recipients will die).

The model reads the N_j time values for group j as input: $0=t_{0j}<t_{1j}<t_{2j}<\ldots<t_{N\,j}$ and the values as $1=v_{0j}>v_{1j}>\ldots>0=v_{Nj}$.

Note that in any given dataset, it would be rare for all recipients to be followed until they die. Some recipients live a long time and some drop out of contact. The user must decide how the survival curve should be extrapolated to $S(t_{Nj}) = 0$ when creating the input file for this step function.

At each graft event, the model samples a value u from a U(0,1) distribution and solves the equation S(T) = u for the survival time T. Solving for the survival time requires putting the sample value, u, into the inverted survival function:

$$T = S^{-1}(u) = S_j^{-1}(\exp(\ln(u)/\exp(\Sigma b_{ij} x_{ij} y_{ij})))$$

To perform this inversion, the model first computes the value of the transformed probability v:

$$v = \exp(\ln(u)/\exp(\Sigma b_{ij} x_{ij} y_{ij}))$$

We are guaranteed that $0 \leq v \leq 1$. The model then uses the step function to determine the corresponding survival time. That is, if the value of v lies between v_{nj} and $v_{n-1,j}$, the survival time is $T = t_{nj}$. This condition will be true for some value of n for n = 1 to N.

The step function for the proportional hazards survival function is the baseline survival curve $s_0(t)$. The step function is a flexible way to approximate survival curves of any shape. Users must remember to enter the baseline survival function[1] rather than the survival function for the average recipient on this screen. In SAS, the baseline survival function can be created as follows:

First, create a separate dataset where all the covariates in the posttransplant survival model are set to zero, e.g.,

```
data covzeros;
  age = 0;  male = 0; weight = 0; height = 0;
run;
```

Then run the posttransplant survival model using this dataset to supply the covariates for the survival curve to be output, e.g.,

```
proc phreg data=analysis_data;
 model recipient_days*died(0) = age male weight height;
baseline out = ph_step_function covariates = covzeros  survival = survival
       /nomeans;
run;
```

The proportional hazards baseline step function data are now saved in the output dataset (ph_step_function in the example above). These data can be entered into the SAM manually or through an input file (Default Survival under Model Structure).

4.2 Time to Graft Failure Determined with a Weibull Survival Model

This section describes the process for entering the parameters which are required in order for KPSAM to determine the time to graft failure using a Weibull survival model.

The Weibull model is one of the exponential families of survival models. A strictly exponential survival model assumes that the same proportion of patients die over the same amount of time, a situation known as having a constant hazard rate. For example, if 10% of the patients die in the first year, then 10% of the remaining 90% will die in the second year, leaving 81% of the original population alive. The parameters entered into the model alter

[1] In the baseline survival function, all covariates in the model are set to zero. This includes covariates that normally cannot equal zero, such as patient weight.

this overall percentage for certain groups of patients, but they do not alter the shape of the curve. For example, older patients may die at a rate of 15% per year while younger patients die at a rate of 5% per year, but in the strictly exponential model both groups are assumed to have this constant mortality rate at all times. Weibull models are more flexible than the strictly exponential models in that their hazard rates can be monotone increasing, decreasing, or constant. This helps model situations like post-transplant mortality, where the hazard is expected to be high at first (due to the operative stresses), then to decrease over time. Aside from the covariates in the model, there are two parameters that define the overall shape of the survival curve, defining this change in the hazard. These two parameters can differ depending on the model formulation.

There are three Weibull model formulations, each corresponding to a commonly-used parameterization of the two Weibull parameters needed. The parameterization to be used depends on the software used to generate the model. For example, SAS output from the lifereg procedure includes the intercept, scale, and shape parameters. In SAS, the scale and the shape parameter are the inverse of each other, so only the shape and the intercept parameters are used in the SAM. If the results of a lifereg procedure in SAS are to be used to describe post-transplant mortality, the formulation used should be the Accelerated Failure Time formulation: "AFT: exp[-exp(shape*(ln(t)-intercept))]". The "shape" and "intercept" should be supplied in the spaces provided. To use another formulation, such as the proportional hazards "PH: exp[-(t/scale)^shape]", use the shape parameter as before and put exp(intercept) in the space provided for the scale. The answers received via each formulation will be identical if the scale and the shape parameters are correctly entered.

4.3 Outcomes

Associated with each survival time in the step function is a table of probabilities for various possible outcomes (death, graft failure, or various relisting events). The model samples a value u from a U(0,1) distribution and compares it with the table of probabilities to determine the outcome.

4.4 Relisting

Associated with each relisting outcome is a time before or after failure at which relisting occurs, and a sequence of status updates for the relisted patient. The model schedules a relisting event at the specified time before or after failure, and schedules the status update events. The model does not automatically schedule a death event for the relisted patient. A death event may be the last of the status updates defined for a relisted patient. If the patient hasn’t received a second transplant by that time, then they will die on the waitlist.

4.5 Graft Failure

For the graft failure outcome, a separate calculation is performed, using the non-relist death definitions, to determine a death date after graft failure for the patient. The model schedules the death event at the specified time.

4.6 Death

For the death outcome, the graft failure equates to death. Therefore, the model schedules a death event on the graft failure date.

5 Creating Input Data Files

Creating suitable input files is not a trivial task. A number of problems can arise when data from real-world data sets are fed into the model. For example, if a status update file is created from real data, then candidates who actually underwent transplant will receive no status updates after the date of transplant. Thus, no candidate who actually underwent transplant will ever die or be removed from the list in the model, even if this means that these candidates survive for months or years as status 1 without undergoing transplant. To obtain realistic results, users should devise a method to generate simulated death or removal dates and status changes for candidates who underwent transplant.

5.1 Validating Input Data

KPSAM does not catch inconsistencies in the input data. For instance, if a patient starts with an active wait time longer than total time on the wait list, KPSAM does not check to make sure those are consistent.

5.2 Sharing Input Files Among Users

If input files will be shared among users on different computers, note that the Allocation Run Specification file refers to each data file with its full path. Some editing of the file may be needed if the directory organization or location of the application on the target machine is different from on the source machine.

5.3 General Principles for All Input Files

Selecting Time Frame for Data
The model will simulate new arrivals, status changes, deaths, and transplants beginning with the snapshot date on the waitlist input data. Statistics will be reported only for the time period designated by the start and end dates selected when the model is run. Any changes that occur before the start date are not reported in the summary statistics. However, they are applied to the waiting list to make it current at the start date.

Formatting DateTime Fields
Format: (mm/dd/yyyy hh:mm:ss) As time information is not included in SRTR data, the time portion was randomly generated during the creation of the supplied input files.

Format of Input Files
All input files are ASCII text files, one line per record, with fields separated by a vertical bar delimiter (“|”).

6 Input File Specifications

This section gives the specifications of the input files. Each specification lists fields in the order they occur in a record, followed by an example. Optional fields are italicized. Non-italicized fields are required. The specification of each field gives the name of the field, its data type, and a short definition. (The field names are those used in the source code.) Files are listed using generic filenames; a particular release may have more specific filenames related to release parameters (described in the Input Files Notes for the release).

6.1 Organ Arrivals

The organs file (organs.txt) contains organs that become available from the model start date through and including the model end date. For inclusion, we suggest selecting records using dispositions 5 (recovered for transplant but not transplanted) and 6 (transplanted).

Sort order: EventTime ascending.

Field Name	Data Type	Short Definition
EventTime	DateTime	organ arrival date-time (mm/dd/yyyy hh:mm:ss)
ItemID	String	item identifier (donor ID)
InstitID	String	institution where donated
DonorAge	Double	age of donor (same units as in patient record)
ABOType	String	ABO blood type of donor (codes defined in blood compatibility file)
HLA A1	Integer	Donor's first A-locus antigen
HLA A2	Integer	Donor's second A-locus antigen
HLA B1	Integer	Donor's first B-locus antigen
HLA B2	Integer	Donor's second B-locus antigen
HLA Dr1	Integer	Donor's first Dr-locus antigen
HLA Dr2	Integer	Donor's second Dr-locus antigen
HLA C1	Integer	Donor's first C-locus antigen
HLA C2	Integer	Donor's second C-locus antigen
HLA DQ1	Integer	Donors first DQ-locus antigen
HLA DQ2	integer	Donor's second DQ-locus antigen
HasKidney	Boolean	Arrival includes a kidney (True|False)
HasSecKidney	Boolean	Arrival includes a second kidney (True|False)
HasPancreas	Boolean	Arrival include a pancreas (True|False)
WeederFld(1)	Double	first referenced weeder field
…		
WeederFld(n)	Double	nth referenced weeder field

WeedFlag(1)	Boolean	first referenced weed flag field
…		
WeedFlag(n)	Boolean	nth referenced weed flag field
OptOrganFld(1)	Specified	first specified optional organ data field
…		
OptOrganFld(n)	Specified	nth specified optional organ data field

Example: 01/02/2008
3:01:01|UL5189|360|21.0|O|3|11|37|49|13|17|6|7|2|3|True|True|False|…

Note: The number of weeder fields expected is determined by what is found in the Optional Data Definition file. The order in which they appear here must be the same as the order in which they appear in the Optional Data Definition file. The number and order of weed flag fields are also determined by what is found in the Optional Data Definition file. The number of optional fields expected, and their order, is also determined by what is found in the Optional Data Definition file. Please refer to the section in this chapter which describes that input file for more information.

6.2 Candidate Waiting List and Candidate Arrivals Files

The Waiting List (waitlist.txt) and Patient Arrivals (patients.txt) files are of the same format. They are both described in this section.

Waitlist.txt contains all candidates on the waiting list (active and inactive) as of the day before the model start.

Patients.txt contains new patients added to the waiting list with listing dates from the model start date through and including the model end date.

The time portion of the EventTime (hh:mm:ss) had to be randomly set since that information is not in the SRTR data. Sometimes a status change occurs on the same day as a new candidate record. As we want all status changes to occur after the new candidate listing, we intentionally set all the new candidate listings to take place in the AM and the status changes in the PM to insure that all status changes submitted on the same day as a new candidate listing occur after the new listing.

Sort order: EventTime ascending.

To avoid over-counting, new candidates who are relisted after undergoing transplant during the model run time period should be excluded from the input data. Relistings after transplant are simulated in the model.

Field Name	**Data Type**	**Short Definition**
SnapDate	DateTime	as-of date for current status of patient (mm/dd/yyyy hh:mm:ss)
EventTime	DateTime	patient arrival date-time (mm/dd/yyyy hh:mm:ss)
ItemID	string	item identifier (patient ID)
CenterID	string	transplantation center identifier
OrgPatAge	double	patient age as of listing {years}
CurPatAge	double	patient age as of snapdate for initial waitlist file {years} patient age at time of listing for patient arrivals file {years}
ABOType	string	ABO blood type (codes defined in blood compatibility file)
HLA-A1	integer	Patient's first A-locus antigen
HLA-A2	integer	Patient's second A-locus antigen
HLA-B1	integer	Patient's first B-locus antigen
HLA-B2	integer	Patient's second B-locus antigen
HLA-Dr1	integer	Patient's first Dr-locus antigen
HLA-Dr2	integer	Patient's second Dr-locus antigen
NeedKidney	boolean	This patient is a candidate for a kidney
NeedPancreas	boolean	This patient is a candidate for a pancreas
OrgKidnStat	string	Original kidney status {1|7|0}
CurKidnStat	string	Current kidney status {1|7|0}
OrgPancStat	string	Original pancreas status {1|7|0}
CurPancStat	string	Current pancreas status {1|7|0}
PRAScore	double	Panel reactive antibody score
WillTakeKid	boolean	Is KP patient willing to accept isolated kidney?
WillTakePan	boolean	Is KP patient willing to accept isolated pancreas?
TotalActWait	double	Elapsed time (days) in active status on wait list as of SnapDate
KidneyWait	double	Elapsed time (days) in active status for kidney as of SnapDate
PancreasWait	double	Elapsed time (days) in active status for pancreas as of SnapDate
InactWait	double	Elapsed time (days) as inactive on the waitlist
WeederFld(1) Min	double	first referenced weeder field minimum value
WeederFld(1) Max	double	first referenced weeder field maximum value
…		
WeederFld(n) Min	double	nth referenced weeder field minimum value
WeederFld(n) Max	double	nth referenced weeder field maximum value
WeedFlag(1)	boolean	first referenced weed flag field
…		
WeedFlag(n)	boolean	nth referenced weed flag field
OptPatientFld(1)	specified	first specified optional patient data field

...		
OptPatientFld(n)	specified	nth specified optional patient data field

Example: (long line has been wrapped)

```
12/31/2000 0:00:03|12/09/1994
3:02:53|5584|527|58.529774127|64|O|11|32|44|56|7|13|True|False|7|7|0|0|6|False
|False|1272|1272|0|943|
0|99|M|White|NonHispanic|1|23.183673469|0|6|0
```

Note: Create the waitlist input file to be fixed as the existing waitlist on the day before the Model Allocation Run start date; i.e., if Model Allocation Run is to start on 1/1/1999, create waitlist ending 12/31/1998. Cumulative times are calculated up to and including the last status change date (SnapDate). For candidates without status changes, cumulative times will all be 0 (see exception below) and SnapDate will be the listing date/time. Waitlist input file sort order: none. Patient Arrivals input file sort order: EventTime ascending.

The number of weeder fields expected is determined by what is found in the Optional Data Definition file. The order in which they appear here must be the same as the order in which they appear in the Optional Data Definition file. The number and order of weed flag fields are also determined by what is found in the Optional Data Definition file. The number of optional fields expected, and their order, is also determined by what is found in the Optional Data Definition file. Please refer to the section in this chapter, which describes that input file for more information.

6.3 Status Updates

The simulation uses actual waitlist arrivals, status changes, and removal information for candidates on the waiting list during the model run timeframe. Status.txt contains records that describe the medical status updates for candidates who have a change in status from the time of the candidate's arrival to the model (either the initial wait list snapshot or arrival to the wait list) until the candidate's removal or death.

We had to randomly set the EventTime (hh:mm:ss) since that information is not in the SRTR data. Sometimes a status change occurs on the same day as a new patient record. As we want status changes to occur after the new patient listing, we intentionally set all the new patients listings to take place in the AM and the status changes to take place in the PM during the creation of the input files. That way if a status change comes through on the same day as a listing, we know it will happen afterward. If more that one status change occurs on the same date, we have not added logic to evaluate their order.

Input data should include all status changes, from the model start date through and including the model end date. Use current Status codes. Status changes should *not* **include those removed due to cadaveric transplant (status 4). Candidates removed due to death will have Status 8. Candidates removed due to a living donor transplant**

will have Status 15. All other status changes that removed the candidate from the waitlist should be set = 9, e.g.., due to improvement in or deterioration of the candidate's condition.

Patients who in real life actually received cadaveric transplants will not receive any status updates after the date of their transplant unless their status histories are completed in some manner, such as the predictive mean matching described in the input files manual. Thus, no patient who actually received a cadaveric transplant will ever die or get removed from the list in the model, even if this means these patients survive for months or years without getting a transplant. In order to obtain realistic results, generate artificial status changes and death or removal dates for patients who received transplants.

Sort order: EventTime ascending. Note: CurKidneyStat 7=inactive; 8=death; 9=removed for reason other than living donor transplant; 15 removed due to living donor transplant

Field Name	Data Type	Short Definition
EventTime	DateTime	event date-time (mm/dd/yyyy hh:mm:ss)
ItemID	string	item ID (Patient ID)
NeedKidney	Boolean	Patient is a candidate for a kidney transplant
NeedPancreas	Boolean	Patient is a candidate for a pancreas transplant
CurKidneyStat	string	Current kidney status {1|7|8|9|0}
CurPancreas-Stat	string	Current pancreas status {1|7|8|9|0}
OptStatusFld(1)	specified	first specified optional status data field
…		
OptStatusFld(n)	specified	nth specified optional status data field

Example: 06/28/2001 13:58:04|149282|True|False|7

Note: The number of optional fields expected, and their order, is determined by what is found in the Optional Data Definition file. Please refer to the section in this chapter which describes that input file for more information.

6.4 Location Mapping

LocMap.txt is a listing of all transplantation center ID's with a code for their related local and regional areas.

Sort order: CenterID unique, ascending. *Note: in the input data supplied, Center ID is masked for confidentiality.*

Field Name	**Short Definition**
CenterID	transplantation center ID / institution where donated
LocalID	local unit ID where center/institution is located
RegionID	region ID where center/institution is located

Example:
7|ALOB|3
9|AROR|3
14|AZGC|5

6.5 Blood Compatibility

ABOChart.txt is a listing of all possible donor and candidate blood type combinations. It consists of an ASCII text file, one line per blood type, "|" delimiter separated fields.

Field Name	**Data Type**	**Short Definition**
OrgABOType	string	Organ ABO blood type {O|A|B|AB}
PatABOType	string	Patient ABO blood type {O|A|B|AB}
CompatCode: Zero mm	string	Code to use for zero-mismatch allocation{I = Identical; C = Compatible; X = Incompatible }
CompatCode: non-zero mm		Code to use for non-zero-mismatch allocation{I = Identical; C = Compatible; X = Incompatible }

Example:
O|O|I|I
O|A|C|X
O|A1|C|X
O|A2|C|X

6.6 Default Data Definition

DefDataDef.txt contains the definition of the default fields that are available for forming allocation rules, score boost specifications, acceptance calculations, and post-graft survival calculations. The program must contain code to support each field that is referenced in this file.

DefDataDef.txt is an ASCII text file that contains blocks, which are surrounded by defined block begin and block end tags.

Field List: <FIELDLIST>…</FIELDLIST>

The field list tag is the outer-most tag. All other blocks are contained inside the field list start and end tags.

Field Definition: <FIELDDEF>...</FIELDDEF>

The field definition block defines one field. The format for this block is:

```
<FIELDDEF>
fldname|fldsource|fldstate|fldtype|fldusage|fldbasis
</FIELDDEF>
```

fldname provides the name of the field. This must be a recognized field name.

fldsource identifies the source {Patient; Organ} of the field.

fldstate must be one of the following {Default; DefCalc; DefComp; DefCut} which defines how the program determines the value of the field.

Default: a field on the default patient or organ input record
DefCalc: a field calculated by default by the program
DefComp: a field whose value is determined by the program by comparing the patient and organ records
DefCut: a field calculated by the program using cutpoints on a known field.

fldtype must be one of the following: {String; Double; DateTime} which defines how the program stores the value of the field.
fldusage must be one of the following: {Category; Score} which defines how the field is used by the program.

fldbasis identifies the numeric field to which the cutpoints are applied if the field state is a cutpoint field. Otherwise, this item is omitted.

Level Definition: <LEVELDEF>...</LEVELDEF>

The level definition block defines the acceptable values for the field. The format for this block is:

```
<LEVELDEF>
label|lowcut|highcut
</LEVELDEF>
```

label identifies, for category fields, which string values can be used for allocation, boost, etc.

lowcut defines the lower cutpoint for this level. Used only for cutpoint (fldstate = DefCut) fields. Otherwise, this item is omitted

highcut defines the upper cutpoint for this level. Used only for cutpoint (fldstate = DefCut) fields. Otherwise, this item is omitted.

Example:
```
<FIELDLIST>
<FIELDDEF>
ABOCompat|Patient|DefComp|String|Category
<LEVELDEF>
I
C
X
</LEVELDEF>
</FIELDDEF>
</FIELDLIST>
```

6.7 Optional Data Definition

OptDataDef.txt consists of three sections: weeder definitions, flag definitions, and optional data definitions.

OptDataDef.txt is an ASCII text file that contains blocks, which are surrounded by defined block-begin and block-end tags.

Weeder Definition: <WEEDDEF>…</WEEDDEF>

Each weeder definition specifies a value field in the organ record and a pair of range fields (minimum and maximum) in the patient (and initial wait list) record. Patients for whom the organ value does not fall within the specified range are weeded from the allocation for that organ. The weeder definition block may define multiple weeder fields. The format for this block is:

```
<WEEDDEF>
fieldname|organidx
</WEEDDEF>
```

fieldname provides the name of the field. The name has to be unique within the data source.
organidx identifies the organ to which this weeder field applies. For the Kidney-Pancreas Model, 1 indicates kidney, 2 indicates pancreas, and 3 indicates both.

Flag Field Definition: <FLAGDEF>…</FLAGDEF>

Each flag field definition specifies a boolean value in the organ record that indicates whether a condition is present and a boolean value in the patient record that indicates whether presence of that condition in an offered organ is acceptable to the patient. Thus, if the organ value is true and the patient value is false, the patient will be weeded from the allocation for that organ. In all other cases, the patient will not be weeded. The flag field definition block may define multiple flag fields. The format for this block is:

```
<FLAGDEF>
fieldname|organidx
</FLAGDEF>
```

fieldname provides the name of the field. The name has to be unique within the data source.
organidx identifies the organ to which this weeder field applies. For the Kidney-Pancreas Model, this should be set to 1 for kidney, 2 for pancreas, and 3 for both.

Optional Data Field Definitions Group: <DATADEF>…</DATADEF>

The Optional Data Field Definitions Group block contains definitions for the optional data fields, each of which is contained in its own <FIELDDEF> block. The format for the optional data definitions group block is:

```
<DATADEF>
<FIELDDEF> … </FIELDDEF>
…
</DATADEF>
```

There are three types of data field definitions which may be defined in the Optional Data Field Definitions Group. They are the Optional Data Field, the Optional Cutpoint Field, and the Optional Calculation Field. Each of the three types is described below. The following definitions apply to all three optional field definitions.

fldname provides the name of the field. The name has to be unique within the data source.
fldsource identifies the source {Patient; Organ; Status} of the field. Patient fields apply, also, to waitlist records, which have the same format as patient records.
fldtype must be one of the following: {String|Double|DateTime} which defines how the program stores the value of the field.

Optional Data Field Definition: <FIELDDEF>…</FIELDDEF>

The optional field definition block defines one optional field. The value Optional indicates to the program that the field will be supplied on the appropriate input file.

The field source parameter specifies which file(s) will include the field – organ, patient and waitlist, or status update. The level definition block within the field definition block contains a list of possible values of Category fields. The format for the optional data field definition is:

```
<FIELDDEF>
fldname|fldsource|Optional|fldtype|fldusage
<LEVELDEF>              { when fldusage=Category }
value-1
...
value-n
</LEVELDEF>
</FIELDDEF>
```

fldusage must be one of the following: {Category|Score|PassThru} which defines how the field is used by the program.
Category identifies a category field. This must be a string.
Score identifies a numeric field. This must be Double or DateTime.
PassThru identifies a field which is read from the input stream and written to the output without being used by the program.

Optional Cutpoint Field Definition: <FIELDDEF>...</FIELDDEF>

When the field state is *OptCut*, a category field will be created by the program using the specified cut points on the specified basis field, **fldbasis**. The level definition block contains a list of possible values for this field. The **label** gives a name to values of the basis field which fall between the lower and upper cutpoints (**lowcut**, **highcut**) defined for this level.

```
<FIELDDEF>
fldname|fldsource|OptCut|fldtype|Category|fldbasis
<LEVELDEF>
label-1|lowcut|highcut
...
label-n|lowcut|highcut
</LEVELDEF>
</FIELDDEF>
```

Optional Calculation Field Definition: <FIELDDEF>...</FIELDDEF>

When the field state is *OptCalc*, a score field will be created by the program using the specified calculation operation.

```
<FIELDDEF>
fldname|fldsource|OptCalc|fldtype|Score
```

```
<OPTCALC>
optfunc|source:variable|source:variable|operator
</OPTCALC>
</FIELDDEF>
```

The calculation specification (<OPTCALC> block) for an OptCalc field consists of one of the following functions:

EnumerNum|source:variable which converts the levels of a categorical variable into a number 0,1,... based upon the order in which the levels of that field were specified. *Source* must be {Patient; Organ}.

Sum|source:variable|source:variable which adds the values of the first and second variables. *Source* must be {Patient; Organ, Literal}, where Literal indicates that a numeric value will be specified instead of a field.

Difference|source:variable|source:variable which subtracts the values of the second variable from the value of the first variable. *Source* must be {Patient; Organ, Literal}.

Ratio|source:variable|source:variable which divides the value of the first (score) variable by the value of the second (score) variable. *Source* must be {Patient; Organ, Literal}.

Product|source:variable|source:variable which multiplies the value of the first (score) variable by the value of the second (score) variable. *Source* must be {Patient; Organ, Literal}.

CompareNum|source:variable|source:variable|operator which compares the values of the two specified variables using the specified comparison operator, returning 1=True or 0=False.
In these calculation specifications, *Source* must be {Patient; Organ; Literal}, as above. *Operator* must consist of one of the following comparison operators {GE; GT; LE; LT; EQ; NE}

NaturalLog|source:variable which gives the natural log of the value of the specified variable. *Source* must be {Patient; Organ, Literal}.

Exponential|source:variable which gives the natural log of the value of the specified variable. *Source* must be {Patient; Organ, Literal}.

Power|source:variable|source:variable which gives the value of the first variable to the power of the value of the second variable. *Source* must be {Patient; Organ, Literal}.

BoolOp|source:variable|source:variable|operator which gives a boolean calculation of the value of the two specified variables. The calculation performed depends on the value of the operator. *Operator* must consist of one of the following Boolean operators {OR, XOR, AND} *Source* must be {Patient; Organ, Literal}.

Equation|equation which gives the result of the specified mathematics formula.

The mathematics formula can be constructed using the elements:

+: operand, executes adding operation
-: operand, executes subtraction operation
*: function, executes multiplying operation
/: function, executes division operation
Sqrt: functions, root of a number. Root can have any degree
Div: functions, executes integer division operation
Mod: functions, executes remainder operation
Int: function, returns the integer part of a number
Frac: function, returns the fractional part of a number
Random: function, returns random number within the range 0 <= value < 1
Trunc: function, truncates a number to an integer
Round: function, returns the value rounded to the nearest whole number
Sin: function, returns the sine of the angle in radians
ArcSin: function, returns the inverse sine of a number
Sinh: function, returns the hyperbolic sine of an angle
ArcSinh: function, returns the inverse hyperbolic sine of a number
Cos: function, returns the cosine of the angle in radians
ArcCos: function, returns the inverse cosine of a number
Cosh: function, returns the hyperbolic cosine of an angle
ArcCosh: function, returns the inverse hyperbolic cosine of a number
Tan: function, returns the tangent of the angle
ArcTan: function, returns the arctangent of a number
Tanh: function, returns the hyperbolic tangent of an angle
ArcTanh: function, the inverse hyperbolic tangent of a number
CoTan: function, returns the cotangent of the angle
ArcCoTan: function, returns the inverse cotangent of a number
CoTanh: function, returns the hyperbolic cotangent of an angle
ArcCoTanh: function, the inverse hyperbolic cotangent of a number
Sec: function, returns the secant of an angle
ArcSec: function, returns the inverse secant of a number
Sech: function, returns the hyperbolic secant of an angle
ArcSech: function, returns the inverse hyperbolic secant of a number
Csc: function, returns the cosecant of an angle
ArcCsc: function, returns the inverse cosecant of a number
Csch: function, returns the hyperbolic cosecant of an angle
ArcCsch: function, returns the inverse hyperbolic secant of a number

Abs: function, returns an absolute value
Ln: function, returns the natural log of an expression
Lg: function, returns log base 10
Log: function, returns the log of expression for a specified base
Pi: function, returns 3.1415926535897932385
Exp: function, returns the exponential of an expression
!: function, returns factorial of an expression
^: function, raises expression to any power
ArcTan2 [Y, X: Double] function, calculates ArcTan(Y/X), and returns an angle in the correct quadrant. The values of X and Y must be between –2^64 and 2^64. Inaddition, the value of X can't be 0. The return value will fall in the range from -Pi to Pi radians.
Hypot [X, Y: Double] function, returns the length of the hypotenuse of a right triangle. Specify the lengths of the sides adjacent to the right angle in X and Y. Hypot usesthe formula Sqrt(X**2 + Y**2)
RadToDeg function, converts radians to degrees
RadToGrad function, converts radians to grads
RadToCycle function, converts radians to cycles
DegToRad function, returns the value of a degree measurement expressed in radians
DegToGrad function, returns the value of a degree measurement expressed in grads
DegToCycle function, returns the value of a degree measurement expressed in cycles
GradToRad function, converts grad measurements to radians
GradToDeg function, converts grad measurements to degrees
GradToCycle function, converts grad measurements to cycles
CycleToRad function, converts an angle measurement from cycles to radians
CycleToDeg function, converts an angle measurement from cycles to degrees
CycleToGrad function, converts an angle measurement from cycles to grads.
LnXP1 function, returns the natural log of (X+1)
Log10 function, calculates log base 10
Log2 function, calculates log base 2
IntPower [Base: Double; Exponent: Integer] function, calculates the integral power of a base value
Power [Base: Double; Exponent: Double] function, Raises Base to any power
Ldexp [X: Double; P: Double] function, calculates X times (2 to the power of P)
Ceil function, rounds variables up toward positive infinity
Floor function, rounds variables toward negative infinity
Poly [X: Double; Coefficients(1)..Coefficients(N): Double] function, evaluates a uniform polynomial of one variable at the value X
Mean [Data(1)..Data(N): Double] function, returns the average of all values in an array
Sum [Data(1)..Data(N): Double] function, returns the sum of the elements in an array
SumInt [Data(1)..Data(N): Integer] function, returns the sum of the elements in an integer array
SumOfSquares [Data(1)..Data(N): Double] function, returns the sum of the squared values from a data array

MinValue [Data(1)..Data(N): Double] function, returns smallest signed value in an array
MinIntValue [Data(1)..Data(N): Integer] function, returns the smallest signed value in an integer array
Min [A,B: Double] function, returns the lesser of two numeric values
MaxValue [Data(1)..Data(N): Double] function, returns the largest signed value in an array
MaxIntValue [Data(1)..Data(N): Integer] function, returns the largest signed value in an integer array
Max [A,B: Double] function, returns the greater of two numeric values
StdDev [Data(1)..Data(N): Double] function, returns the sample standard deviation for elements in an array
PopnStdDev [Data(1)..Data(N): Double] function, calculates the population standard deviation
Variance [Data(1)..Data(N): Double] function, calculates statistical sample variance from an array of data
PopnVariance [Data(1)..Data(N): Double] function, calculates the population variance
TotalVariance [Data(1)..Data(N): Double] function, returns the statistical variance from an array of values
Norm [Data(1)..Data(N): Double] function, returns the Euclidean 'L-2' norm
RandG [Mean, StdDev: Double] function, generates random numbers with Gaussian distribution
RandomRange [AFrom, ATo: Integer] function, returns a random integer from a specified range
RandomFrom [Value(1)..Value(N): Double] function, returns a randomly selected element from an array
EnsureRange [AValue, AMin, AMax: Double] function, returns the closest value to a specified value within a specified range

Example:

```
<WEEDDEF>
DonorAge|3
</WEEDDEF>
#
<FLAGDEF>
HCVDonor|3
</FLAGDEF>
#
<DATADEF>
<FIELDDEF>
PatientGender|Patient|Optional|String|Category
<LEVELDEF>
M
F
</LEVELDEF>
```

```
</FIELDDEF>
<FIELDDEF>
PatientRace|Patient|Optional|String|Category
<LEVELDEF>
Amerind
Arabian
Asian
Black
Hawaiian
Indian
White
</LEVELDEF>
</FIELDDEF>
<FIELDDEF>
WtRatio|Patient|OptCalc|Double|Score
<OPTCALC>
Ratio|Organ:DonorWeight|Patient:PatientWeight
</OPTCALC>
</FIELDDEF>
<FIELDDEF>
AdultNum|Patient|OptCalc|Double|Score
<OPTCALC>
CompareNum|Patient:PatientAge|Literal:18|GE
</OPTCALC>
</FIELDDEF>
<FIELDDEF>
PediatNum|Patient|OptCalc|Double|Score
<OPTCALC>
CompareNum|Patient:PatientAge|Literal:18|LT
</OPTCALC>
</FIELDDEF>
<FIELDDEF>
WtRatA|Patient|OptCalc|Double|Score
<OPTCALC>
CompareNum|Patient:WtRatio|Literal:0.7|GT
</OPTCALC>
</FIELDDEF>
<FIELDDEF>
AgeDeltaabs|Patient|OptCalc|Double|Score
<OPTCALC>
Equation|Abs (Patient:(“can_age”) - Organ:(“don_age”))
</OPTCALC>
</FIELDDEF>
</DATADEF>
```

6.8 Allocation Method Definition

DefMethods.txt contains a list of allocation method definitions, consisting of a list of default rule definitions. Within a rule definition, the <GROUDEF> block defines the subset of organs for which the rule applies. Fields used in the group definition must be category fields that are defined in either the default data definition file or the optional data definition file. Group components (lines in the groupdef section) are joined using a logical AND. When an organ arrives, the program checks the allocation methods in the order in which they are presented to find the first one that specifies a group to which the organ belongs. If the organ characteristics don't match any of the groups, the organ is discarded and an error message is written to the input stream errors output file.

The lines in the <ORDERDEF> block specify subsets of the patient list. The order in which the order definitions appear determines the order in which subsets of the waiting list are handled. Within a subset, patients are sorted by the sortfields.

Rule List: <RULELIST>…</RULELIST>

The rule list tag is the outer-most tag. All other blocks are contained inside the rule list start and end tags.

Allocation Definition: <ALLOCDEF>…</ALLOCDEF>

The allocation definition block defines the allocation rules for one group of organs. There may be multiple allocation definition blocks within the rulelist block. The format for the allocation definition block is:

```
<ALLOCDEF>
title
<GROUPDEF>
source:fieldname=level
…
</GROUPDEF>
<ORDERDEF>                    these groups | are sorted this way
patfield=level,…|sortfield,…
patfield=level,…|sortfield,…
…
</ORDERDEF>
</ALLOCDEF>
```

source:fieldname=level defines the group to which an organ belongs, based on its characteristics. The source is always organ. The field name is any field in the default or optional data definition files that apply to the organ. Level is the value this field must have in order for this organ to be in this group. The organ must characteristics must match all of the definitions in the group.

patfield=level defines a subset of patients to whom this order definition applies.

sortfield defines the sort order of the subset of patients who meet the patient criteria of this order definition statement.

Example:

```
<RULELIST>
<ALLOCDEF>
OPTN Kidney Allocation System. ABO Type O, Age < 35 Donors
<GROUPDEF>
Organ:HasKidney=True
Organ:HasPancreas=False
Organ:IsPayBack=False
Organ:DonAgeRngCat2=under35
Organ:ABOType=O
</GROUPDEF>
<ORDERDEF>
CurKidnStat=1,ABOCompat=I,GeogLevel=LOCAL,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOCompat=I,Sensitized=True,OPOIsOwedTwo=True,IsZeroMiss=True|DebtTime,KidneyPoints
CurKidnStat=1,ABOCompat=I,Sensitized=True,GeogLevel=REGXLOC,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOCompat=I,Sensitized=True,GeogLevel=NATXREG,IsZeroMiss=True|KidneyPoints
CurKid-
nStat=1,ABOCompat=I,Sensitized=False,CurAgeGroup=Pediat,OPOIsOwedTwo=True,IsZeroMiss=True|DebtTime,KidneyPoints
CurKidnStat=1,ABOCompat=I,Sensitized=False,CurAgeGroup=Pediat,GeogLevel=REGXLOC,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOCompat=I,Sensitized=False,CurAgeGroup=Pediat,GeogLevel=NATXREG,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOCompat=I,Sensitized=False,OPOIsOwedTwo=True,IsZeroMiss=True|DebtTime,KidneyPoints
CurKidnStat=1,ABOCompat=I,Sensitized=False,GeogLevel=REGXLOC,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOCompat=I,Sensitized=False,GeogLevel=NATXREG,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOType=B,GeogLevel=LOCAL,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOType=B,Sensitized=True,OPOIsOwedTwo=True,IsZeroMiss=True|DebtTime,KidneyPoints
CurKidnStat=1,ABOType=B,Sensitized=True,GeogLevel=REGXLOC,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOType=B,Sensitized=True,GeogLevel=NATXREG,IsZeroMiss=True|KidneyPoints
CurKid-
nStat=1,ABOType=B,Sensitized=False,CurAgeGroup=Pediat,OPOIsOwedTwo=True,IsZeroMiss=True|DebtTime,KidneyPoints
CurKidnStat=1,ABOType=B,Sensitized=False,CurAgeGroup=Pediat,GeogLevel=REGXLOC,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOType=B,Sensitized=False,CurAgeGroup=Pediat,GeogLevel=NATXREG,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOType=B,Sensitized=False,OPOIsOwedTwo=True,IsZeroMiss=True|DebtTime,KidneyPoints
CurKidnStat=1,ABOType=B,Sensitized=False,GeogLevel=REGXLOC,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOType=B,Sensitized=False,GeogLevel=NATXREG,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOCompat=C,GeogLevel=LOCAL,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOCompat=C,Sensitized=True,OPOIsOwedTwo=True,IsZeroMiss=True|DebtTime,KidneyPoints
CurKidnStat=1,ABOCompat=C,Sensitized=True,GeogLevel=REGXLOC,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOCompat=C,Sensitized=True,GeogLevel=NATXREG,IsZeroMiss=True|KidneyPoints
CurKid-
nStat=1,ABOCompat=C,Sensitized=False,CurAgeGroup=Pediat,OPOIsOwedTwo=True,IsZeroMiss=True|DebtTime,KidneyPoint
s
CurKidnStat=1,ABOCompat=C,Sensitized=False,CurAgeGroup=Pediat,GeogLevel=REGXLOC,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOCompat=C,Sensitized=False,CurAgeGroup=Pediat,GeogLevel=NATXREG,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOCompat=C,Sensitized=False,OPOIsOwedTwo=True,IsZeroMiss=True|DebtTime,KidneyPoints
CurKidnStat=1,ABOCompat=C,Sensitized=False,GeogLevel=REGXLOC,IsZeroMiss=True|KidneyPoints
CurKidnStat=1,ABOCompat=C,Sensitized=False,GeogLevel=NATXREG,IsZeroMiss=True|KidneyPoints
```

```
CurKid-
nStat=1,ABOIdentOrCompat=True,Sensitized=False,OPOHighDebt=True,GeogLevel=NATWREG,IsZeroMiss=True|InvTotDebt,Ki
Points
CurKid-
nStat=1,ABOIdentOrCompat=True,GeogLevel=LOCAL,IsZeroMiss=False,WithinAcptMM=True|EnumHighRank,PedAtListing,KiP
oints
CurKid-
nStat=1,ABOIdentOrCompat=True,GeogLevel=REGXLOC,IsZeroMiss=False,WithinAcptMM=True|EnumHighRank,PedAtListing,
KiPoints
CurKid-
nStat=1,ABOIdentOrCompat=True,GeogLevel=NATXREG,IsZeroMiss=False,WithinAcptMM=True|EnumHighRank,PedAtListing,
KiPoints
</ORDERDEF>
</ALLOCDEF>
</RULELIST>
```

6.9 Allocation Score Boost Definition

DefBoostDef.txt is used to specify rules that define how scores used to sort patients might be boosted. The idea is that if the offered organ meets specified criteria (listed in the <ORGGRP> block) and the patient meets specified criteria (listed in the <PATGRP> block), then the value of that patient's score will be boosted in the defined manner (defined in the <BOOST> block) for this particular organ being allocated. The boost consists of multiplying the existing score by a value and then adding a second value. Boost definitions may only reference the fields that are described in either the default data definition file or the optional data definition file.

Boost List: <BOOSTLIST>…</BOOSTLIST>
The boost list tag is the outer-most tag. All other blocks are contained inside the boost list start and end tags.

Boost Definition: <BOOSTDEF>…</BOOSTDEF>

The boost definition block contains the definition of one boost rule. There may be multiple boost rules contained within the boost list block.

There are two possible formats for the boost definition block. The format for the standard boost definition block is:

```
<BOOSTDEF>
boostname
<ORGGRP>
fldname=level
…
</ORGGRP>
<PATGRP>
```

```
    fldname=level
    ...
    </PATGRP>
    <BOOST>
    fldname|boostamnt|boostfact
    ...
    </BOOST>
    </BOOSTDEF>
```

boostname: gives the name of this boost rule
fldname=level: specifies the criteria the organ and patient must meet in order for the patient to get this boost.
fldname: within the <BOOST> block specifies the field in the patient record which will get the new, booostedscore, where boostedscore = **fldname** * **boostfact** + **boostamnt**

Example:

```
<BOOSTLIST>
<BOOSTDEF>
Zero Dr Antigen Mismatch Kidney Points
<ORGGRP>
HasKidney=True
</ORGGRP>
<PATGRP>
CntDrMiss=Zero
</PATGRP>
<BOOST>
KidneyPoints|2|1|365.25|0|1
</BOOST>
</BOOSTDEF>
</BOOSTLIST>
```

The alternate boost definition block allows the user to define a linear equation by which to boost the underlying variable. Note that this alternate boost definition can only be entered through input files. There are no input panels in KPSAM from which the inputs may be made.

The format for the alternate, linear equation, boost definition block is:

```
<BOOSTDEF>
boostname
<ORGGRP>
fldname=level
...
```

```
</ORGGRP>
<PATGRP>
fldname=level
...
</PATGRP>
<CALCDEF>
fldname
<TERMDEF>
coeff1|source1a:fieldname1a|source1b:fieldname1b
...
</TERMDEF>
</CALCDEF>
</BOOSTDEF>
```

boostname: gives the name of this boost rule
fldname: specifies the field in the patient record which will get the new, boostedscore, where boostedscore = **fldname** *operator* (coeff1*source1a:fieldname1a*source1b:fieldname1b) *operator* ... *operator* (coeff(i)*source(i)a:fieldname(i)a*source(i)b:fieldname(i)b)

operator: the optional operator field is used to indicate the mathematical operation to perform. The default is Sum. Other option is Product. The example below yields this equation:

eNLSB=eNLSB + (21.7258 + (-0.1944 * can_age) + (-0.1852*can_age*can_dgn_kp_dm)) * 0.8*one_minus_DPI_rank

```
<BOOSTLIST>
<BOOSTDEF>
Life Years From Transplant (LYFT) Equation
<CALCDEF>
eNLSB
<TERMDEF>
21.7258
-0.1944|Patient:can_age
-0.1852|Patient:can_age|Patient:can_dgn_kp_dm
Product|0.8|Organ:one_minus_DPI_rank
</TERMDEF>
</CALCDEF>
</BOOSTLIST>
</BOOSTDEF>
```

6.10 Default Acceptance

DefAccept.txt is the default definition of calculation to perform to determine whether a patient accepts an offered organ.

Acceptance List: <ACCLIST>…</ACCLIST>

The acceptance list tag is the outer-most tag. All other blocks are contained inside the acceptance list start and end tags.

Calculation Definition: <CALCDEF>…</CALCDEF>

Each calculation definition block defines the calculation that will take place, the mathematical function that will be used, and the characteristics of the organ and/or the patient which will be inputs to the mathematical function. The format for the calculation definition block is:

```
<CALCDEF>
accname
<GROUPDEF>
source:fieldname=level
...
</GROUPDEF>
<TERMDEF>
coeff|source1:fieldname1|source2:fieldname2
...
</TERMDEF>
<FUNCTION>
funcname
</FUNCTION>
</CALCDEF>
```

accname is the name of this acceptance calculation.

Fields used to define groups must be category fields that are defined in either the default data definition file or the optional data definition file. Group components (lines in the <GROUPDEF> block) are joined using a logical AND.
source:fieldname=level: lists the numeric fields associated with the patient and the organ which will be inputs to the mathematical function. Source: {Patient, Organ}.

Fields used to define terms must be score fields that are defined in either the default data definition file or the optional data definition file. Each term component (line in the <TERMDEF> block) consists of the coefficient (**coeff**) times the first field (**source1:fieldname1**) times the second field (**source2:fieldname2**). Term components (lines in the <TERMDEF> block) are added together. If the second field of a term component is omitted, then the term component consists of the coefficient times the first field. If both fields are omitted, then the term component consists of just the coefficient.

The function section identified the function to be applied to the result of the linear combination specified in the termdef section. The inverse logit function, InvLogit(βX) = exp(βX) / 1+exp(βX), where βX = the result of the linear combination specified in the termdef section, is currently the only choice available. Specify INVLOGIT for the **funcname**.

Example:

```
<ACCLIST>
<CALCDEF>
Sample Acceptance
<GROUPDEF>
</GROUPDEF>
<TERMDEF>
3
0.02|Patient:PatientAge
-0.04|Organ:DonorAge
</TERMDEF>
<FUNCTION>
INVLOGIT
</FUNCTION>
</CALCDEF>
</ACCLIST>
```

6.11 Default Post-Graft Survival Definition

DefSurvival.txt is the default definition of the calculation to perform to determine how long before graft failure or death after transplant and to assign a series of possible outcomes, such as relisting and status changes, to that patient prior to graft failure.

Survival List: <SURVLIST>…</SURVLIST>

The survival list tag is the outer-most tag. All other blocks are contained inside the survival list start and end tags.

Calculation Definition: <CALCDEF>…</CALCDEF>

Each calculation definition block defines the characteristics of the organ and/or the patient to which this survival calculation applies. The format for the calculation definition block is:

```
<CALCDEF>
```

```
survname
<GROUPDEF>
source:fieldname=level
...
</GROUPDEF>
<TERMDEF>
coeff|source1:fieldname1|source2:fieldname2
...
</TERMDEF>
<FUNCTION>
funcname
<STEPFUNC>
<STEP>
value|time
<OUTCOMES>
outtype|outcomename|outprob|outtime
<UPDATES>
atime|needkid|needpanc|kidstat|pancstat|opt(1)|...
...
</UPDATES>
...
</OUTCOMES>
</STEP>
...
</STEPFUNC>
</FUNCTION>
</CALCDEF>
```

survname is the name of this survival calculation.

Fields used to define groups must be category fields that are defined in either the default data definition file or the optional data definition file. Group components (lines in the <GROUPDEF> block) are joined using a logical AND.
source:fieldname=level: lists the numeric fields associated with the patient and the organ which will be inputs to the mathematical function. Source: {Patient, Organ}.

Fields used to define terms must be score fields that are defined in either the default data definition file or the optional data definition file. Each term component (line in the <TERMDEF> block) consists of the coefficient (**coeff**) times the first field (**source1:fieldname1**) times the second field (**source2:fieldname2**). Term components (lines in the <TERMDEF> block) are added together. If the second field of a term component is omitted, then the term component consists of the coefficient times the first field. If both fields are omitted, then the term component consists of just the coefficient.

The <FUNCTION> block identifies the type of function to be used to determine graft survival time. The choices for **funcname** are STEPFUNC or WEIBULL. If **funcname** is StepFunc, then the next block will be <STEPFUNC>. If **funcname** is Weibull, then the next block will be <WEIBULL>

The <STEPFUNC> block identifies the step function to be inverted to determine graft survival time:

$\text{Prob[survival to time} > \text{T]} = S(T)^{\exp(y)}$

where y = the result of the linear combination specified in the termdef section.

The <STEPFUNC> block defines the function as a set of steps connecting the listed points. Associated with each point is a **value** (probability) and a **time** (in days) until failure.

The <WEIBULL> block identifies the values of the parameters and form of a Weibull equation.

parm1|parm2|form name

The parameters required depend on the chosen form. For example, if the chosen form is "Intercept" then the required parameters are shape and intercept. The following table outlines the Weibull equation form choices and their required parameters.

Weibull Equation form	**parm1**	**parm2**	**form name**
AFT: exp[-exp(shape*(ln(t)-intercept))]	Shape	intercept	Intercept
PH: exp[-scale * t^{shape}]	Shape	scale	Lambda
PH: exp[-(t/scale)^{shape}]	Shape	scale	Alpha

Whether the graft failure date is determined by a Cox PH StepFunction or a Weibull Distribution, the patient outcomes prior to failure are determined by a series of steps which depend on the days to failure.

Each <STEP> block refers to patients with a certain number of days until graft failure. Within each step, there can be one or more outcomes.

Each <OUTCOME> block includes a probability (**outprob**) and a time (**outtime**, in days) before failure that the outcome occurs. For a death outcome, the time is zero. For a relist outcome, the time is the number of days before failure that relist occurs. For a failure outcome, the time is zero. This outcome triggers the non-relist death mechanism to determine a death date. Non-relist death is described in the following section.

Associated with each relist outcome is a set of status updates (<UPDATES> block) for the relisted patient. The relisted patient updates consist of a time (**atime**, in days) after relist that the update occurs. The set of updates should include an update for death, since only the graft failure event will have been generated from the failure time for the step. The format of <UPDATES> block is the same as that for the Status Updates input file, which is described earlier in this chapter, except that the patient id is not specified since these post-graft survival status updates may be applied to any patient who receives a transplant.

Example:

```
<SURVLIST>
<CALCDEF>
Kidney Recipients
<GROUPDEF>
Patient:GotKidney=True
Patient:GotPancreas=False
</GROUPDEF>
<TERMDEF>
0.5895138812|Patient:rec_age_lt02
-0.868481171|Patient:rec_age_0210
-1.007488554|Patient:rec_age_1117
-0.549892487|Patient:rec_age_1834
0.00347155|Organ:cod_donor_anoxia
0.0820965508|Organ:cod_donor_stroke
-0.003845639|Organ:cod_donor_cns
-0.020776126|Organ:cod_donor_oth
</TERMDEF>
<FUNCTION>
StepFunc
<STEPFUNC>
1|0
0.9997155859|1
0.8557388865|836
0.8031129461|2914
</STEPFUNC>
<OUTDIST>
<STEP>
836
<OUTCOMES>
Death|Death|0.5|0
Fail|Graft Failure|0.05|0
Relist|Relist Group 1|0.45|700.0
<UPDATES>
```

```
0.0|TRUE|FALSE|1|0
750.0|TRUE|FALSE|8|8
</UPDATES>
</OUTCOMES>
</STEP>
</OUTDIST>
</FUNCTION>
</CALCDEF>
</SURVLIST>
```

6.12 Non-relist Death

DefSurvival.txt is the default definition of the calculation to perform to determine how long before graft failure or death after transplant and to assign a series of possible outcomes, such as relisting and status changes, to that patient prior to graft failure.

Survival List: <SURVLIST>…</SURVLIST>
The survival list tag is the outer-most tag. All other blocks are contained inside the survival list start and end tags.

Calculation Definition: <CALCDEF>…</CALCDEF>
Each calculation definition block defines the characteristics of the organ and/or the patient to which this survival calculation applies. The format for the calculation definition block is:

```
<CALCDEF>
survname
<GROUPDEF>
source:fieldname=level
…
</GROUPDEF>
<TERMDEF>
coeff|source1:fieldname1|source2:fieldname2
…
</TERMDEF>
<FUNCTION>
funcname
<STEPFUNC>
<STEP>
value|time
<OUTCOMES>
outtype|outcomename|outprob|outtime
<UPDATES>
```

```
atime|needkid|needpanc|kidstat|pancstat|opt(1)|...
...
</UPDATES>
...
</OUTCOMES>
</STEP>
...
</STEPFUNC>
</FUNCTION>
</CALCDEF>
```

survname is the name of this survival calculation.

Fields used to define groups must be category fields that are defined in either the default data definition file or the optional data definition file. Group components (lines in the <GROUPDEF> block) are joined using a logical AND.
source:fieldname=level: lists the numeric fields associated with the patient and the organ which will be inputs to the mathematical function. Source: {Patient, Organ}.

Fields used to define terms must be score fields that are defined in either the default data definition file or the optional data definition file. Each term component (line in the <TERMDEF> block) consists of the coefficient (**coeff**) times the first field (**source1:fieldname1**) times the second field (**source2:fieldname2**). Term components (lines in the <TERMDEF> block) are added together. If the second field of a term component is omitted, then the term component consists of the coefficient times the first field. If both fields are omitted, then the term component consists of just the coefficient.

The <FUNCTION> block identifies the type of function to be used to determine graft survival time. The choices for **funcname** are STEPFUNC or WEIBULL. If **funcname** is StepFunc, then the next block will be <STEPFUNC>. If **funcname** is Weibull, then the next block will be <WEIBULL>

The <STEPFUNC> block identifies the step function to be inverted to determine graft survival time:

Prob[survival to time > T] = $S(T)^{\exp(y)}$
where y = the result of the linear combination specified in the termdef section.

The <STEPFUNC> block defines the function as a set of steps connecting the listed points. Associated with each point is a **value** (probability) and a **time** (in days) until failure.

The <WEIBULL> block identifies the values of the parameters and form of a Weibull equation.

```
parm1|parm2|form name
```

The parameters required depend on the chosen form. For example, if the chosen form is "Intercept" then the required parameters are shape and intercept. The following table outlines the Weibull equation form choices and their required parameters.

Weibull Equation form	parm1	parm2	form name
AFT: exp[-exp(shape*(ln(t)-intercept))]	Shape	intercept	Intercept
PH: exp[-scale * t^{shape}]	Shape	scale	Lambda
PH: exp[-$(t/scale)^{shape}$]	Shape	scale	Alpha

Whether the graft failure date is determined by a Cox PH StepFunction or a Weibull Distribution, the patient outcomes prior to failure are determined by a series of steps which depend on the days to failure.

Each < DefSurvival.txt is the default definition of the calculation to perform to determine how long before graft failure or death after transplant and to assign a series of possible outcomes, such as relisting and status changes, to that patient prior to graft failure.

Survival List: <SURVLIST>…</SURVLIST>
The survival list tag is the outer-most tag. All other blocks are contained inside the survival list start and end tags.

Calculation Definition: <CALCDEF>…</CALCDEF>
Each calculation definition block defines the characteristics of the organ and/or the patient to which this survival calculation applies. The format for the calculation definition block is:

```
<CALCDEF>
survname
<GROUPDEF>
source:fieldname=level
…
</GROUPDEF>
<TERMDEF>
coeff|source1:fieldname1|source2:fieldname2
…
</TERMDEF>
<FUNCTION>
funcname
<STEPFUNC>
<STEP>
value|time
```

```
<OUTCOMES>
outtype|outcomename|outprob|outtime
<UPDATES>
atime|needkid|needpanc|kidstat|pancstat|opt(1)|...
...
</UPDATES>
...
</OUTCOMES>
</STEP>
...
</STEPFUNC>
</FUNCTION>
</CALCDEF>
```

survname is the name of this survival calculation.

Fields used to define groups must be category fields that are defined in either the default data definition file or the optional data definition file. Group components (lines in the <GROUPDEF> block) are joined using a logical AND.
source:fieldname=level: lists the numeric fields associated with the patient and the organ which will be inputs to the mathematical function. Source: {Patient, Organ}.

Fields used to define terms must be score fields that are defined in either the default data definition file or the optional data definition file. Each term component (line in the <TERMDEF> block) consists of the coefficient (**coeff**) times the first field (**source1:fieldname1**) times the second field (**source2:fieldname2**). Term components (lines in the <TERMDEF> block) are added together. If the second field of a term component is omitted, then the term component consists of the coefficient times the first field. If both fields are omitted, then the term component consists of just the coefficient.

The <FUNCTION> block identifies the type of function to be used to determine graft survival time. The choices for **funcname** are STEPFUNC or WEIBULL. If **funcname** is StepFunc, then the next block will be <STEPFUNC>. If **funcname** is Weibull, then the next block will be <WEIBULL>

The <STEPFUNC> block identifies the step function to be inverted to determine graft survival time:

Prob[survival to time > T] = $S(T)^{\exp(y)}$
where y = the result of the linear combination specified in the termdef section.

The <STEPFUNC> block defines the function as a set of steps connecting the listed points. Associated with each point is a **value** (probability) and a **time** (in days) until failure.

The <WEIBULL> block identifies the values of the parameters and form of a Weibull equation.

parm1|parm2|form name

The parameters required depend on the chosen form. For example, if the chosen form is “Intercept” then the required parameters are shape and intercept. The following table outlines the Weibull equation form choices and their required parameters.

Weibull Equation form	parm1	parm2	form name
AFT: exp[-exp(shape*(ln(t)-intercept))]	Shape	intercept	Intercept
PH: exp[-scale * t^{shape}]	Shape	scale	Lambda
PH: exp[-$(t/scale)^{shape}$]	Shape	scale	Alpha

Whether the graft failure date is determined by a Cox PH StepFunction or a Weibull Distribution, the patient outcomes prior to failure are determined by a series of steps which depend on the days to failure.

Each <STEP> block refers to patients with a certain number of days until graft failure. Within each step, there can be one or more outcomes.

Each <OUTCOME> block includes a probability (**outprob**) and a time (**outtime**, in days) before failure that the outcome occurs. For a death outcome, the time is zero. For a relist outcome, the time is the number of days before failure that relist occurs. For a failure outcome, the time is zero. This outcome triggers the non-relist death mechanism to determine a death date. Non-relist death is described in the following section.

Associated with each relist outcome is a set of status updates (<UPDATES> block) for the relisted patient. The relisted patient updates consist of a time (**atime**, in days) after relist that the update occurs. The set of updates should include an update for death, since only the graft failure event will have been generated from the failure time for the step. The format of <UPDATES> block is the same as that for the Status Updates input file, which is described earlier in this chapter, except that the patient id is not specified since these post-graft survival status updates may be applied to any patient who receives a transplant.

Example:
<SURVLIST>
<CALCDEF>
Kidney Recipients
<GROUPDEF>
Patient:GotKidney=True

```
Patient:GotPancreas=False
</GROUPDEF>
<TERMDEF>
0.5895138812|Patient:rec_age_lt02
-0.868481171|Patient:rec_age_0210
-1.007488554|Patient:rec_age_1117
-0.549892487|Patient:rec_age_1834
0.00347155|Organ:cod_donor_anoxia
0.0820965508|Organ:cod_donor_stroke
-0.003845639|Organ:cod_donor_cns
-0.020776126|Organ:cod_donor_oth
</TERMDEF>
<FUNCTION>
StepFunc
<STEPFUNC>
1|0
0.9997155859|1
0.8557388865|836
0.8031129461|2914
</STEPFUNC>
<OUTDIST>
<STEP>
836
<OUTCOMES>
Death|Death|0.5|0
Fail|Graft Failure|0.05|0
Relist|Relist Group 1|0.45|700.0
<UPDATES>
0.0|TRUE|FALSE|1|0
750.0|TRUE|FALSE|8|8
</UPDATES>
</OUTCOMES>
</STEP>
</OUTDIST>
</FUNCTION>
</CALCDEF>
</SURVLIST>
```

6.13 Non-relist Death

DefNRDeath.txt is the default definition of the calculation to perform to determine how long after graft failure a graft recipient who does not relist will live.

Non-relist Death List: <NRDEATHLIST>...</NRDEATHLIST>
The non-relist death list tag is the outer-most tag. All other blocks are contained inside the non-relist death list start and end tags.

Calculation Definition: <CALCDEF>...</CALCDEF>
Each calculation definition block defines the characteristics of the organ and/or the patient to which this survival calculation applies. The format for the calculation definition block is:

```
<NRDEATHLIST>
<CALCDEF>
calcname
<GROUPDEF>
source:fieldname=level
#...
</GROUPDEF>
<TERMDEF>
coeff|source:fieldname|source:fieldname
#...
</TERMDEF>
<FUNCTION>
funcname
<STEPFUNC>
<STEP>
value|time
<STEP>
#...
</STEPFUNC>
</FUNCTION>
</CALCDEF>
#...
</NRDEATHLIST>
```

The contents of this file are very similar in form to those in the DefSurvival.txt file, described previously in this chapter. Please refer to that section for specific information on the contents of this file.

6.14 Partially Met Requirement Relist

DefPartRelist.txt is the default definition of calculation to perform to determine whether a kidney-pancreas patient who receives an isolated kidney or an isolated pancreas will immediately relist for the organ not received.

```
<OUTCOMELIST>
outtype|outname                    // outtype: {Relist}
#...
</OUTCOMELIST>
<PARTIALLIST>
<CALCDEF>
calcname
<GROUPDEF>
source:fieldname=level             // source: {Patient; Organ}
#...
</GROUPDEF>
<TERMDEF>
coeff|source:fieldname|source:fieldname
#...
</TERMDEF>
<FUNCTION>
InvLogitUp                  // funcname: {InvLogitUp}
<STEPFUNC>                  // container for outcomes
<STEP>                             // container for outcomes
value|time                  // container for outcomes
<OUTCOMES>
outtype|outname|outprob|rltime
<UPDATES>
DaysAfterRelist|NeedKidney|NeedPancreas|KidneyStat|PancreasStat
#...
</UPDATES>
</OUTCOMES>
</STEP>
</STEPFUNC>
</FUNCTION>
</CALCDEF>
#...
</PARTIALLIST>
```

The linear combination of term components yields a term y, to which the inverse logit function, InvLogit(y) = Exp(y)/(1+Exp(y)), is applied. This produces the probability that a KP patient with partially met requirements relists for the organ not received.

The step function specification consists of one step, and is used primarily as a container for the outcomes, which in turn are used as containers for status updates. The relist time in the outcome is used to specify the time in days after the graft was received at which the patient relists.

6.15 Initial Kidney Payback Accounting

Paybacks.txt contains the initial status of payback accounting for kidneys by OPO and ABO type.

```
<PAYBACKS>
<LOCALE>
localid
<PAYITEM>
abotype|debtcount          // negative debt count indicates credits
<CREDITS>
datetime
#...
</CREDITS>
</PAYITEM>
#...
</LOCALE>
#...
</PAYBACKS>
```

Note that if the debt count is negative for a blood type, it represents number of credits, and is followed by a credit block that lists the dates of the credits. If the debt count is positive for a blood type, the OPO owes organs for that blood type, and no credits appear.

6.16 Antigen Splits

AntigenSplit.txt identifies antigen pairs that are not considered a mismatch.

```
PatientAntigen|DonorAntigen

<MATCHLIST>
<LOCUS site="A">
1|1
2|2
203|2
210|2
3|3
#...
</LOCUS>
<LOCUS site="B">
5|5
52|5
53|5
```

```
78|5
#...
</LOCUS>
<LOCUS site="DR">
1|1
103|1
2|2
15|2
16|2
#...
</LOCUS>
</MATCHLIST>
```

6.17 Zero Mismatch

ZeroMM.txt is an optional file. For each donor listed in the file, there is a list of candidates in the given cohort that have no mismatches at the HLA A, B, and DR loci. For each donor present in this file, the processing time required to find the list of zero-mismatch candidates us eliminated. If a donor is not listed, each candidate must be checked by the program. Therefore, having a complete zero mismatch file in place can substantially improve the performance of KPSAM. If a donorID is listed without any patientID's, it is assumed that there are no zero-mismatch candidates in the cohort for this donor. If a given donorID is not present, then all candidates are checked.

```
DonorID|PatientID|PatientID|PatientID ...
```

6.18 Unacceptable Antigen Equivalences

UnAntEquiv.txt identifies antigen pairs that are considered equivalent. For each antigen listed for a given patient in the Patient Unacceptable Antigens file, all of the donor equivalences listed in this Unacceptable Antigen Equivalences table are considered unacceptable, and the candidate will be screened from match runs for donors for which any of these antigens is present. Unacceptable antigens may be listed for the A, B, C, DR, and DQ loci.

```
PatientAntigen|DonorAntigen

<MATCHLIST>
<LOCUS site="A">
1|1
2|2
#...
```

```
</LOCUS>
</MATCHLIST>
```

6.19 Patient Unacceptable Antigen Updates

UnAccAnt.txt updates the patients' lists of unacceptable antigens. These lists can change over time, and this file is the mechanism by which they are updated.

Unacceptable antigens for candidates are entered in sets. The sets are identified by the waitlist registration audit ID. Whenever a new waitlist registration audit ID is encountered for a candidate, the candidate's existing list of unacceptable antigens is deleted, and the new set is entered.

Date & Time Stamp|PatientID|Waitlist Registration Audit ID|HLA Locus|Antigen

6.20 Pediatric Transplant Time Goals

Goals by which time pediatric patients should receive kidney transplant.

Field Name	Data Type	Short Definition
AgeCut	double	original patient age cutpoint (OrgPatAge<AgeCut)
TargetDays	double	goal of maximum waiting time for transplant

Example:

6|182.625	*// patients under 6 years old should be transplanted within ½ year*
11|365.25	*// patients under 11 years old should be transplanted within 1 year*
18|547.875	*// patients under 18 years old should be transplanted within 1½ year*

6.21 Model Allocation Runs

ModelRuns.txt is created by the application, not by the user. The ModelRuns.txt contains the names and file prefixes of all Allocation Runs that have been defined using the application. The model saves the specification when the user hits the OK button or the Run button in the user interface.

Annotated Example: (the annotations in italics do not appear in the input file)

NAME OPTN Test	*name of the Allocation Run*
FILE OPTN	*short name used as file prefix*
EOF	*end of definition*
NAME OldRules Test	*name of another run*
FILE OldRules	*short name used as 2nd file prefix*
EOF	*end of 2nd definition*

6.22 Allocation Run Definition

xxxxModel.inp is generated by the application and is used to save an Allocation Run definition (where xxxx denotes the short name of the Allocation Run). Each file contains the specifications of one of the defined Allocation Runs. The model saves the specification when the user hits the OK button or the Run button in the user interface.

The definitions include paths where the directory can locate the input files for the Allocation Run. If the Allocation Run definition is to be shared among users on different computers, it will be necessary to edit the path definitions to correspond to the locations on the new machine where the application can find the files.

In addition, the allocation run definition file will contain the entire configuration of the allocation run, which will look similar to the contents of the input files previously defined in this chapter.

7 Output File Specifications

This section gives the specifications of output files for the program. The specification of each output file lists fields in the order in which they occur in a record, followed by an example. The format of each table list gives the name of the field, its data type, and a short definition. The field names are those used in the source code. The program assigns each output file a different name by prefixing the short title of the Allocation Run. ***Note:*** *xxxx denotes the Short Name given during the Model Allocation Run in the descriptions below.*

Output Files:

1. Waitlist at end of Model Allocation Run
2. Patients receiving grafts
3. Death list (candidates who died during the Model Allocation Run)
4. Removed list (candidates who were removed from the wait list during the Model Allocation Run)
5. Relisted (candidates who were relisted following a transplant during the Model Allocation Run)
6. Graft list (all of the organs grafted during the Model Allocation Run and the candidate receiving the graft)
7. Screened candidates (log of candidates who were screened from the match run)
8. Match list (all candidates to whom each organ was offered)
9. Status updates (log of the arriving status update events)
10. Event log (log of the arriving events)
11. Payback status (as of the end of the Model Allocation Run)
12. Summary statistics

The following five output files all consist of the same layout. For simplicity, the output layout is only specified one time. All are text files, one line per record, "|" delimiter separated fields.

7.1 Waiting List

xxxxWait.out – list of candidate records that represents the status of the waitlist at the end time of the Model Allocation Run.

7.2 Grafted Patients

xxxxGrpat.out – list of patient records that represents all candidates receiving a graft during the Model Allocation Run. Recipients receiving more than one graft will be in this file multiple times.

7.3 Death Listing

xxxxDeath.out – list of candidate records that represents all patients who died during the Model Allocation Run. It includes waitlist deaths, removed candidate deaths, and post transplant deaths.

7.4 Removed Listing

xxxxRemove.out – list of the patients who were removed from the waitlist during the Model Allocation Run.

7.5 Relistings

xxxxRelist.out – list of the patients who were relisted after transplant during the Model Allocation Run. A patient who has received two transplants during the model run will be listed twice in the relistings file. Each time a patient is relisted, their PatientId is incremented by 0.1, so that relisted patients can be easily identified on the waitlist and in the grafted patients output file.

Field Name	Data Type	Short Definition
Iter	integer	Replication number
SnapDate	DateTime	Snapshot for existing wait list date-time (mm/dd/yyyy hh:mm:ss)
ArrivalTime	DateTime	Date-time arriving or relisting on wait list (mm/dd/yyyy hh:mm:ss)
PatientId	string	Item ID (patient ID)
CenterID	string	Transplantation center ID
OrgPatAge	double	Patient age at time of listing
CurPatAge	double	Patient age
ABOType	string	ABO blood type {O|A|B|AB}
HLA-A1	integer	Patient's first A-locus antigen
HLA-A2	integer	Patient's second A-locus antigen
HLA-B1	integer	Patient's first B-locus antigen
HLA-B2	integer	Patient's second B-locus antigen
HLA-Dr1	integer	Patient's first Dr-locus antigen
HLA-Dr2	integer	Patient's second Dr-locus antigen
NeedKidney	boolean	This patient is a candidate for a kidney
NeedPancreas	boolean	This patient is a candidate for a pancreas
OrgKidnStat	string	Original kidney status {1|7|0}
CurKidnStat	string	Current kidney status {1|7|0}
OrgPancStat	string	Original pancreas status {1|7|0}
CurPancStat	string	Current pancreas status {1|7|0}
PRAScore	double	Panel reactive antibody score

WillTakeKid	boolean	Is KP patient willing to accept isolated kidney?
WillTakePan	boolean	Is KP patient willing to accept isolated pancreas?
CumActTime	double	Elapsed time (days) in active status on wait list as of SnapDate
CumKidneyTime	double	Cumulative time active on kidney waitlist {days}
CumPan-creasTime	double	Cumulative time active on pancreas waitlist {days}
CumInactTime	double	Cumulative time in inactive status{days}
WeederFld(1) Min	double	first referenced weeder field minimum value
WeederFld(1) Max	double	first referenced weeder field maximum value
…		
WeederFld(n) Min	double	nth referenced weeder field minimum value
WeederFld(n) Max	double	nth referenced weeder field maximum value
WeedFlag(1)	boolean	first referenced weed flag field
…		
WeedFlag(n)	boolean	nth referenced weed flag field
OptPatientFld(1)	specified	first specified optional patient data field
…		
OptPatientFld(n)	specified	nth specified optional patient data field
GraftCount	double	Number of grafts, including this one, that this patient has received
FailDate	DateTime	Date-time the patient's graft is scheduled to fail (mm/dd/yyyy hh:mm:ss)
DeathDate	DateTime	Date-time the patient died in the model (mm/dd/yyyy hh:mm:ss)
RelistDate	DateTime	Date-time of patient relisting (mm/dd/yyyy hh:mm:ss)
{GraftID}	string	Organ identifier
{KidneyPoints}	integer	Kidney points calculated for allocation
{ProbAccept}	double	Calculated probability that this patient would accept this or-gan
{GraftDate}	DateTime	Date-time of transplant (mm/dd/yyyy hh:mm:ss)
{PtRlstDate}	DateTime	Date-time patient is scheduled to relist (mm/dd/yyyy hh:mm:ss)
{GotKidney}	boolean	Did this patient receive a kidney during the model run? (True|False)
{GotPancreas}	boolean	Did this patient receive a pancreas during the model run? (True|False)
{NumAMiss}	integer	Number of A-locus antigen mismatches associated with this transplant

{NumBMiss}	integer	Number of B-locus antigen mismatches associated with this transplant
{NumDrMiss}	integer	Number of Dr-locus antigen mismatches associated with this transplant
{AllocRuleName}	string	Name of the allocation rule with which the organ was place
{AllocRuleNum}	integer	0-based index to the line of the allocation rule that placed the organ
(SortField0)	string	Name of the first sort field that was calculated for this placement
{Sort0}	integer	Value that was calculated for the first sort field for this placement
(SortField1)	string	Name of the second sort field that was calculated for this placement
{Sort1}	integer	Value that was calculated for the second sort field for this placement
(SortField2)	string	Name of the third sort field that was calculated for this placement
{Sort2}	integer	Value that was calculated for the third sort field for this placement
(SortField3)	string	Name of the fourth sort field that was calculated for this placement
{Sort3}	integer	Value that was calculated for the fourth sort field for this placement

7.6 Grafted Organs

xxxxGraft.out – list of all organs grafted during the Model Allocation Run and the patient receiving the graft.

Field Name	**Data Type**	**Short Definition**
Iter	integer	Replication number
OrganId	string	Organ ID
EventTime	DateTime	Date-time of graft (mm/dd/yyyy hh:mm:ss)
HasKidney	boolean	This organ package contained a kidney (True|False)
HasSecKid	boolean	This organ package contained a second kidney (True|False)
HasPancreas	boolean	This organ package contained a pancreas (True|False)
OrganInstId	string	Donor Institution ID
DonorAge	double	Donor age
OrganABO	string	Donor ABO blood type {O|A|B|AB}
PatientId	string	Patient ID

ProbAccept	double	Probability of acceptance calculated by the model
GotKidney	boolean	This patient received a kidney (True|False)
GotPancreas	boolean	This patient received a pancreas (True|False)
PatCenterID	string	Transplantation center ID
PatientAge	double	Patient age
PatientABO	string	Patient ABO blood type {O|A|B|AB}
CurKidnStat	string	Kidney status at time of transplant {1|7|0}
CurPancStat	string	Pancreas status at time of transplant {1|7|0}
CumTotTime	double	Cumulative active waiting time {days}
CumKidneyTime	double	Cumulative active waiting time for kidney {days}
CumPancreasTime	double	Cumulative active waiting time for pancreas {days}
CumInactTime	double	Cumulative time in inactive status {days}
ArrivalTime	DateTime	Date-time arriving on the wait list (mm/dd/yyyy hh:mm:ss)

7.7 Screened Candidates

xxxxScreened.out - list of candidates who were screened from the match run for each organ due to a weeder field being out of range or because of an unacceptable antigen. This file is only produced if requested during the Model Allocation Run by marking the Log Screen check box.

Field Name	**Data Type**	**Short Definition**
Iter	integer	Replication number
ScreenReason	String	Description of the reason for screening this candidate from this match
OrganId	string	Organ ID
PatientId	string	Patient ID
AllocRuleName	string	Name of the allocation rule in which this offer was made
AllocRuleEntry	integer	0-based index to the line of the allocation rule for this offer

7.8 Matched Offers

xxxxMatch.out – list of all candidates to whom each organ was offered. This file is only produced if requested during the Model Allocation Run by marking the Log Match check box.

Field Name	**Data Type**	**Short Definition**
Iter	integer	Replication number

OrganId	string	Organ ID
EventTime	DateTime	Organ arrival date-time (mm/dd/yyyy hh:mm:ss)
HasKidney	boolean	This organ package contained a kidney (True|False)
HasSecKid	boolean	This organ package contained a second kidney (True|False)
HasPancreas	boolean	This organ package contained a pancreas (True|False)
OrganInstId	string	Donor Institution ID
DonorAge	double	Donor age
OrganABO	string	Donor ABO blood type {O|A|B|AB}
PatientId	string	Patient ID
OfferNumber	double	How many times this organ has been offered, counting this offer
Decision	boolean	Patient accepted the offered organ (Accepted|Declined|Discarded)
ProbAccept	double	Probability of acceptance calculated by the model
SortField0	string	Name of the first sort field that was calculated for this offer
Sort0	integer	Value that was calculated for the first sort field for this offer
SortField1	string	Name of the second sort field that was calculated for this offer
Sort1	integer	Value that was calculated for the second sort field for this offer
SortField2	string	Name of the third sort field that was calculated for this offer
Sort2	integer	Value that was calculated for the third sort field for this offer
SortField3	string	Name of the fourth sort field that was calculated for this offer
Sort3	integer	Value that was calculated for the fourth sort field for this offer
GotKidney	boolean	This patient received a kidney (True|False)
GotPancreas	boolean	This patient received a pancreas (True|False)
PatCenterID	string	Transplantation center ID
PatientAge	double	Patient age at time of listing
PatientABO	string	Patient ABO blood type {O|A|B|AB}
CurKidnStat	string	Kidney status at time of offer {1|7|0}
CurPancStat	string	Pancreas status at time of offer {1|7|0}
CumTotTime	double	Cumulative active waiting time {days}
CumKidneyTime	double	Cumulative active wait time for kidney {days}
CumPancreasTime	double	Cumulative active wait time for pancreas {days}
CumInactTime	double	Cumulative inactive time on waitlist {days}
AllocRuleName	string	Name of the allocation rule in which this offer was made
AllocRuleEntry	integer	0-based index to the line of the allocation rule for this offer

Status Updates

xxxxStatus.out – log of the arriving status update events. This file is only produced if requested during the Model Allocation Run by marking the Log Updates check box.

Field Name	**Data Type**	**Short Definition**
Iter	integer	Replication number
EventTime	DateTime	Date-time of arrival of the update (mm/dd/yyyy hh:mm:ss)
PatientId	string	Patient ID
NeedKidney	boolean	Patient is a candidate for a kidney transplant
NeedPancreas	boolean	Patient is a candidate for a pancreas transplant
CurKidneyStat	string	Current kidney status {1|7|8|9|0|.}
CurPancStat	string	Current pancreas status {1|7|8|9|0|.}
OptStatusFld(1)	specified	first specified optional status data field
…		
OptStatusFld(n)	specified	nth specified optional status data field

7.9 Event Log

EventLog.log – log of the arriving events. This file is only produced if requested during the Model Allocation Run by marking the Log Events check box.

Field Name	**Data Type**	**Short Definition**
EventTime	DateTime	Date-time of arrival of the update (mm/dd/yyyy hh:mm:ss)
EventID	String	Event ID {PatientID|OrganID}
EventDesc	String	Description of the event

Paybacks Status

xxxxPayback.out – similar to the input paybacks file, this lists the number of paybacks owed by OPO and blood type as of the end data of the model run period.

7.10 Summary

xxxxSummary.out – summary statistics for each iteration, followed by the mean and standard deviation of each statistic across all the iterations in the Allocation Run.

Field Name	**Data Type**	**Short Definition**
Iteration x	Integer	one block of summary statistics for each iteration
Kidney Initial Waitlist	Integer	kidney patients on wait list at start time
Pancreas Initial Waitlist	Integer	pancreas patients on wait list at start time

Kidney-Pancreas Initial Waitlist	Integer	kidney-pancreas patients on wait list at start time
New Kidney Patient Listings	Integer	new kidney patient arrivals between start and end time
New Pancreas Patient Listings	Integer	new pancreas patient arrivals between start and end time
New Kidney-Pancreas Patient Listings	Integer	new kidney-pancreas patient arrivals between start and end time
Removed from Kidney Waitlist for reason other than living donor transplant	integer	kidney patients who were removed from the waitlist between start and end time for reason other than living donor transplant
Removed from Pancreas Waitlist for reason other than living donor transplant	Integer	pancreas patients who were removed from the waitlist between start and end time for reason other than living donor transplant
Removed from Kidney-Pancreas Waitlist for reason other than living donor transplant	Integer	kidney-pancreas patients who were removed from the waitlist between start and end time for reason other than living donor transplant
Removed from Kidney Waitlist due to living donor transplant	Integer	kidney patients who were removed from the waitlist between start and end time due to living donor transplant
Removed from Pancreas Waitlist due to living donor transplant	Integer	pancreas patients who were removed from the waitlist between start and end time due to living donor transplant
Removed from Kidney-Pancreas Waitlist due to living donor transplant	Integer	kidney-pancreas patients who were removed from the waitlist between start and end time due to living donor transplant
Final Kidney Waitlist	Integer	final kidney waitlist at end of run
Final Pancreas Waitlist	Integer	final pancreas waitlist at end of run
Final Kidney-Pancreas Waitlist	Integer	final kidney-pancreas waitlist at end of run
Discarded Kidneys	Integer	number of discarded kidneys
Discarded Pancreas	Integer	number of discarded pancreata
Kidney Transplants	integer	number of patients who received a kidney transplant
Pancreas Transplants	integer	number of patients who received a pancreas transplant
Kidney-Pancreas Transplants	integer	number of patients who received a kidney-pancreas transplant

Kidney Patient Relistings	integer	number of patients who relisted for a kidney after a transplant
Pancreas Patient Relistings	integer	number of patients who relisted for a pancreas after a transplant
Kidney-Pancreas Patient Relistings	integer	number of patients who relisted for a kidney-pancreas after a transplant
Kidney Retransplants	integer	number of patients who received a kidney transplant after another transplant
Pancreas Retransplants	integer	number of patients who received a pancreas transplant after another transplant
Kidney-Pancreas Retransplants	integer	number of patients who received a kidney-pancreas transplant after another transplant
Pediatric Kidney Transplants	integer	number of pediatric kidney transplants
Pediatric Pancreas Transplants	integer	number of pediatric pancreas transplants
Pediatric Kidney-Pancreas Transplants	integer	number of pediatric kidney-pancreas transplants
ABO Identical Kidney Transplants	integer	number of ABO identical kidney transplants
ABO Compatible Kidney Transplants	integer	number of ABO compatible kidney transplants
ABO Identical Pancreas Transplants	integer	number of ABO identical pancreas transplants
ABO Compatible Pancreas Transplants	integer	number of ABO compatible pancreas transplants
ABO Identical Kidney-Pancreas Transplants	integer	number of ABO identical kidney-pancreas transplants
ABO Compatible Kidney-Pancreas Transplants	integer	number of ABO compatible kidney-pancreas transplants
Kidney Waitlist Deaths	integer	deaths of patients while on the kidney waitlist
Pancreas Waitlist Deaths	integer	deaths of patients while on the pancreas waitlist
Kidney-Pancreas Waitlist Deaths	integer	deaths of patients while on the kidney-pancreas waitlist
Isolated Kidney to KP Patient	integer	number of kidney-pancreas patients who received only a kidney transplant
KP Gets Kidney and Relists for Pancreas	integer	number of such patients who relisted for pancreas (subset of total pancreas relistings, above)
Isolated Pancreas to KP Patient	integer	number of kidney-pancreas patients who received only a pancreas transplant
KP Gets Pancreas and Relists for Kidney	integer	number of such patients who relisted for kidney (subset of total kidney relistings, above)

Zero ABDr Mismatch Kidney-Pancreas Transplants	integer	number of KP transplants that were 0-ABDr antigen mismatch
One ABDr Mismatch Kidney-Pancreas Transplants	integer	number of KP transplants that were 1-ABDr antigen mismatch
Zero BDr Mismatch Kidney-Pancreas Transplants	integer	number of KP transplants that were 0-BDr antigen mismatch
One BDr Mismatch Kidney-Pancreas Transplants	integer	number of KP transplants that were 1-BDr antigen mismatch
Zero Dr Mismatch Kidney-Pancreas Transplants	integer	number of KP transplants that were 0-Dr antigen mismatch
One Dr Mismatch Kidney-Pancreas Transplants	integer	number of KP transplants that were 1-Dr antigen mismatch
Zero ABDr Mismatch Isolated Kidney Transplants	integer	number of kidney transplants that were 0-ABDr antigen mismatch
One ABDr Mismatch Isolated Kidney Transplants	integer	number of kidney transplants that were 1-ABDr antigen mismatch
Zero BDr Mismatch Isolated Kidney Transplants	integer	number of kidney transplants that were 0-BDr antigen mismatch
One BDr Mismatch Isolated Kidney Transplants	integer	number of kidney transplants that were 1-BDr antigen mismatch
Zero Dr Mismatch Isolated Kidney Transplants	integer	number of kidney transplants that were 0-Dr antigen mismatch
One Dr Mismatch Isolated Kidney Transplants	integer	number of kidney transplants that were 1-Dr antigen mismatch
Zero ABDr Mismatch Isolated Pancreas Transplants	integer	number of pancreas transplants that were 0-ABDr antigen mismatch
One ABDr Mismatch Isolated Pancreas Transplants	integer	number of pancreas transplants that were 1-ABDr antigen mismatch
Zero BDr Mismatch Isolated Pancreas Transplants	integer	number of pancreas transplants that were 0-BDr antigen mismatch
One BDr Mismatch Isolated Pancreas Transplants	integer	number of pancreas transplants that were 1-BDr antigen mismatch
Zero Dr Mismatch Isolated Pancreas Transplants	integer	number of pancreas transplants that were 0-Dr antigen mismatch
One Dr Mismatch Isolated Pancreas Transplants	integer	number of pancreas transplants that were 1-Dr antigen mismatch

8 Frequently Asked Questions

8.1 Data Input Paramaters

Model Allocation Run Dates: 1/1/2010 through 1/1/2011
Organs available between 1/1/2010 and 12/31/2010
Wait List statistics as of 12/31/2009
New Patients added between 1/1/2010 and 12/31/2010
Status Updates between 1/1/2010 and 12/31/2010

8.2 Summary Report – Standard Deviation Calculation

Each iteration represents an independent sample from an infinite set of possible iterations. The variance around each statistic is thus given by the sum of the squared differences from the mean, divided by number of iterations minus one. The standard deviation is the square root of the variance.

8.3 Do Transplanted Organs Include Retransplants?

Yes. It counts all organs that are transplanted, whether it was the patient's first transplant or not.

8.4 Do Patient Listings Include Relistings?

No. Patient Listings includes only the initial listing.

8.5 Should the inputs and outputs balance?

Yes. This is the initial list + all entries to the list - all exits from the list.

```
  Initial Wait List
+ Patient Listings
+ Relistings
- Transplanted Organs
- Removed from Wait List
- Wait List Deaths
- Relisted Patient Deaths
====================
Final Wait List
```

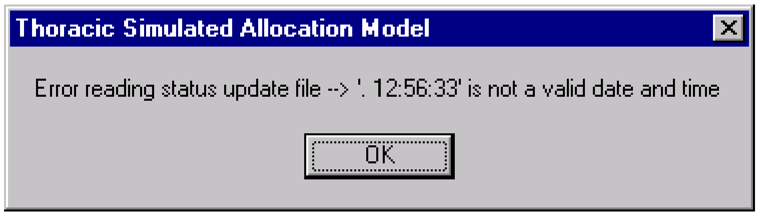
8.6 What records are included in Graft.out vs. Grpat.out?

Graft.out contains a record for each organ transplanted. Grpat.out also contains one record for each transplant that occurs during the model run, but it contains information about the patient as well as the organ.

8.7 What are the types of Input/Output Errors?

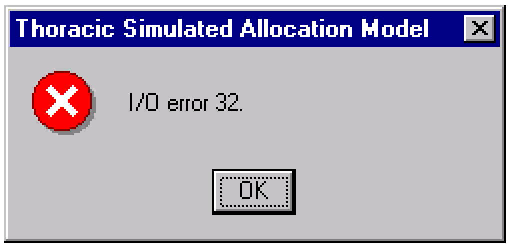
Typically the KPSAM program will not run with errors in the input files. Below are the most frequently encountered errors. Although these illustrations use the dialog boxes from the thoracic simulated allocation model, the KPSAM versions will display the same messages.

If a value read from a text file does not conform to the expected format, an error message will be displayed during the model allocation run specifying the location of the error. The input data will have to be corrected and the model re-started.

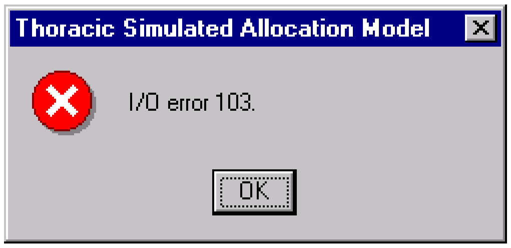

Thoracic Simulated Allocation Model

Error reading status update file --> '. 12:56:33' is not a valid date and time

If an output file is in use by another program, i.e., Excel or Word, you will get “I/O error 32.” Close the file in use and restart the model run. If a file you have selected in your input parameters does not exist, you will get “I/O error 103.”


Thoracic Simulated Allocation Model

I/O error 32.


Thoracic Simulated Allocation Model

I/O error 103.

9 Additional Resources

OPTN maintains a glossary of terminology related to transplantation: http://optn.transplant.hrsa.gov/resources/glossary.asp

SRTR maintains summary information on organ allocation policy, including flowcharts and descriptive articles: http://www.srtr.org/allocationcharts/Default.aspx

The KPSAM data input files are based on the SRTR Standard Analysis File (SAF), and the SAF data dictionary may be useful in understanding these fields in more detail: http://www.srtr.org/data_request/saf.aspx.

Requests for SAF data may be made to the SRTR: http://www.srtr.org/data_request/datause_policy.aspx

Please send questions to SRTR: srtr@srtr.org or 1-877-970-SRTR.